ICON-2018

15th International Conference on Natural Language Processing

Proceedings of the Conference

15-18 December 2018
Punjabi University, Patiala, India

Preface

Natural Language Processing (NLP) research has evolved from the era of punch cards and batch processing, in which the analysis of a sentence could take several minutes, to the era of Google and the likes of it, in which millions of webpages can be processed in matter of milliseconds. NLP enables computers to perform a wide range of natural language related tasks at all levels, ranging from parsing and opinion mining, to machine translation and speech recognition. Research in Natural Language Processing (NLP) has taken a noticeable leap in the recent years. Tremendous growth of information on the web and its easy access has stimulated large interest in the field. With the ongoing growth of the World Wide Web and social media, there is a drastic increase in online data. As the amount of data increases the mechanisms to process these unstructured data and to extract meaningful information from it becomes more challenging. These challenges and difficulties can be overcome with the advanced NLP techniques. Deep learning architectures and algorithms are making impressive advances in NLP. Recent NLP research is now increasingly focusing on the use of new deep learning methods.

Indic language processing presents formidable challenges to achieving multilingualism and multiculturalism in the Indian subcontinent. One of the first and most obvious challenges is the multitude and diversity of languages: India is a land of diverse culture with around 33 major languages and 1,652 dialects from half-a-dozen different language groups. Indic Language Processing involves developing software in Indic Scripts/ languages, Input methods, Localization of computer applications, Web development, Internationalized Domain Name (IDN), OCR, Spell-checkers, Speech applications etc. Research on Indian language technology has thrived in the past few years, with ICON (International Conference on Natural Language Processing), being the premier conference in this field.

This volume contains papers selected for presentation in technical sessions of 15th International Conference on Natural Language Processing (ICON-2018) and short communications selected for poster presentation. We are thankful to our excellent team of reviewers from all over the globe who deserve full credit for the hard work put in for reviewing the high quality submissions with rich technical content. From 152 submissions, 30 papers have been finally selected, 16 for oral and 14 for poster presentation, representing a variety of new and interesting developments, covering a wide spectrum of NLP areas and core linguistics.

We are deeply grateful to Dr. Pushpak Bhattacharyya, Director and Professor of Computer Science and Engineering, IIT Patna, India and Mr. Sanjeev Gupta, Flipkart, Bangalore, for delivering the keynote lectures at ICON-2018. We would also like to thank the members of the Advisory Committee and Programme Committee for their support and co-operation in making ICON 2018 a success.

We thank Prof. Sanjay Dwivedi and Dr. Sarika Jain, Chair, Student Paper Competition and Dr. M D Kulkarni and Dr. Rajeev R R, Chair, NLP Tools Contest for taking the responsibilities of the events.

We are also grateful to Dr. Anil Kumar Singh for devoting his precious time to shape this volume in its present form. His timely help enable us to release this volume in time.

We convey our thanks to Balwant Singh, Dharwinder Singh, Rakesh Dawra from Punjabi University Patiala and P V S Ram Babu from International Institute of Information Technology (IIIT), Hyderabad for their dedicated efforts in successfully handling the ICON Secretariat. We heartily express our

gratitude to the dedicated team of volunteers at Punjabi University Patiala and the entire staff of Research Centre for Technical Development of Punjabi Language Literature and Culture, Punjabi University Patiala for working tirelessly day and night for the success of the conference.

We also thank all those who came forward to help us in this task. We apologize if we have missed some names.

Finally, we thank all the researchers who responded to our call for papers and all the participants of ICON-2018, without whose overwhelming response the conference would not have been a success.

December 2018
Patiala

Gurpreet Singh Lehal
Dipti Misra Sharma
Rajeev Sangal

Conference General Chair:

Rajeev Sangal, IIT (BHU), Varanasi, India

Programme Committee:

Kalika Bali, Microsoft Research, India
Srinivas Bangalore, Interactions LLC, AT&T Research, USA
Rajesh Bhatt, University of Massachusetts, USA
Pushpak Bhattacharyya, IIT Patna, India
Monojit Choudhury, Microsoft Research, India
Harald Hammarström, Max Planck Institute for Psycholinguistics, Nijmegen, The Netherlands
Mohammed Hasanuzzaman, Université de Caen, Normandie, France
Jyoti Pawar, DCST, Goa University, India
L Ramamoorthy, CIIL Mysore, India
Owen Rambow, University of Columbia, USA
Elizabeth Sherly, IIITM-K,Trivandrum, India
Keh-Yih Su, Institute of Information Science, Academia Sinica, Taiwan
Vasudeva Varma, IIIT Hyderabad, India

NLP Software Chairs:

M. D. Kulkarni, CDAC Pune, India (Chair)
Rajeev R.R., ICFOSS Thiruvananthapuram, India (Co-Chair)

Student Paper Competition Chair:

Sanjay Dwivedi, BBAU (Central University Lucknow), India (Chair)
Sarika Jain, NIT Kurukshetra, India (Co-Chair)

Organizing Committee:

Vishal Goyal, Punjabi University Patiala, India
Gurpreet Singh Joshan, Punjabi University Patiala, India
Tejinder Singh Saini, Punjabi University Patiala, India
Ankur Rana, Punjabi University Patiala, India

Sponsors

Microsoft® Research

Flipkart 

विज्ञान एवं प्रौद्योगिकी विभाग
DEPARTMENT OF
SCIENCE & TECHNOLOGY

Referees

We gratefully acknowledge the excellent quality of refereeing we received from the reviewers. We thank them all for being precise and fair in their assessment and for reviewing the papers in time.

Adithya Pratapa
Ajit Kumar
Amol Bole
Anchal Rani
Ankur Rana
Anoop Kunchukuttan
Anshul Bawa
Anupam Jamatia
Anupam Mondal
Anushiya G
ANZI A S
Arjun Akula
Arun Baby
Aswin Shanmugam
Atul Ojha
Avijit Thawani
Bankim Patel
Basant Agarwal
Bhubvaneshwari Melinamath
Braja Patra
C Chandra Shekhar
Deepali Singla
Delia Hernández Farias
Devi G
Dhanya K
Dinesh Kumar
Dr. B. Pawar
Dr.Preeti Dubey
Dwijen Rudrapal
Golda Rajan
Gurpreet Josan
Gurpreet Lehal
K Mishra
Kamal Garg
Kamal Sarkar
Kevin Patel
Kunal Chakma
Latha Nair
Lovely Sharma
Mamoon Rashid
Mayank Singh
Niloofar Safi
Nirmal Surange
Parminder Singh
Parteek Bhatia
Partha Pakray
Pranav Dhakras
Pranav Goel
Prateek Agrawal
Rajeev Puri
Rajiv Srivastava
Ravinder Kumar
Rudramurthy V
Sachin Pawar
Sachin Ruikar
Sanjeev Sharma
Santanu Pal
Shanky Goel
Shashank Gupta
Shilpa Desai
Shubhnandan Jamwal
Sivanand Achanta
Sopan Kolte
Sourabh Kumar
Sourav Mandal
Subba Oota
Sudipta Kar
Tejinder Singh Saini
Thoudam Singh
Umamaheswari E
Umrinder Singh
V Singh
Vasudeva Varma
Venkata Viraraghavan
Vijay Ram
Vijayalakshmi P
Vimal Soni
Vishal Goyal
Vishal Gupta
Yan Shao

Table of Contents

Conference Program

Sunday, December 16, 2018

+ 8:00-9:30 Registration

+ 9:30-11:00 Inaugural Ceremony

+ 9:30-11:00 Keynote Lecture 1 by Dr. Pushpak Bhattacharya

+ 11:00-11:30 Tea Break

+ 11:30-13:15 Technical Session I : Speech and Summarization

Abstractive Summarization Using Attentive Neural Techniques
Jacob Krantz and Jugal Kalita

Challenges and Issues in Developing an Annotated Corpus and HMM POS Tagger for Khasi
Medari Janai Tham

Infant Crying Cause Recognition using Conventional and Deep Learning based Approaches
Shivam Sharma, P. Viswanath and Vinay Kumar Mittal

Automatic Pause Boundary and Pause Duration Detection for Text-to-Speech Synthesis Systems in Indian Languages
Atish Shankar Ghone, Rachana Nerpagar, Pranaw Kumar, BiraChandra Singh, Prakash B. Pimpale and Sasikumar M.

+ 13:15-14:15 Lunch

Sunday, December 16, 2018 (continued)

+ 14:15-15:30 Technical Session II : NLP Tools

Summarization of Table Citations from Text
Monalisa Dey, Salma Mandi and Dipankar Das

Khasi Shallow Parser
Medari Janai Tham

Resolving Actor Coreferences in Hindi Narrative Text
Nitin Ramrakhiyani, Swapnil Hingmire, Sachin Pawar, Sangameshwar Patil, Girish K. Palshikar, Pushpak Bhattacharyya and Vasudeva Verma

Deep Learning methods for Semantic Role Labeling in Indian Languages
Aishwary Gupta, Akshay Pawale and Manish Shrivastava

Knowledge Building via optimally clustered Word Embedding with Hierarchical Clustering
Sandaru Seneviratne, Rangika Samarawickrama, Nadeesha Pathirana, Shane Wolff, Charith Chitraranjan, Uthayasanker Thayasivam and Tharindu Ranasinghe

+ 15:30-15:45 Tea Break

+ 15:45-17:45 Poster Session

Machine Learning Approaches for Amharic Parts-of-speech Tagging
Ibrahim Gashaw and H. L. Shashirekha

English to Bodo Statistical Machine Translation System Using Multi-domain Parallel Corpora
Saiful Islam

POS Tagging and Named Entity Recognition on Handwritten Documents
Vijay Rowtula and Praveen Krishnan

A New Chat Solution for Shared Services using Natural Language Processing Models
M. Saravanan, Satheesh K. Perepu and Sudipta Bose

Does Curriculum Learning help Deep Learning for Natural Language Generation?
Sandhya Singh, Kevin Patel, Pushpak Bhattacharya, Krishnanjan Bhattacharjee, Hemant Darbari and Seema Verma

Sunday, December 16, 2018 (continued)

WupLeBleu: The Word-net Based Evaluation Metric for Machine Translation
Debajyoty Banik, Asif Ekbal and Pushpak Bhattacharyya

"Is This A Joke?": A Large Humor Classification Dataset
Faraz Faruqi and Manish Shrivastava

Fuzzy Evolutionary Self-Rule generation and Text Summarization
Pradeepika Verma and Hari Om

A Content-based Recommendation System for Medical Concepts: Disease and Symptom
Anupam Mondal, Dipankar Das and Sivaji Bandyopadhyay

A Deep Learning Model for Event Extraction and Classification in Hindi for Disaster Domain
Zishan Ahmad, Sahoo Sovan Kumar, Asif Ekbal and Pushpak Bhattacharyya

Exploring the importance of context and embeddings in neural NER models for task-oriented dialogue systems
Pratik Jayarao, Chirag Jain and Aman Srivastava

Improving Computer Generated Dialog with Auxiliary Loss Functions and Custom Evaluation Metrics
Thomas Conley , Jack St. Clair and Jugal Kalita

Analyzing Autism Speech of Children in English Vowels Regions by Analysis of Changes in Production Features
Abhijit Mohanta and Vinay Kumar Mittal

+ 18:00-19:00 NLPAI Meeting

Sunday, December 16, 2018 (continued)

+ 19:00-20:30 Cultural Programme

+ 20:30-Onwards Dinner

Monday, December 17, 2018

+ 9:30-10:15 Industry Session Talk by Sanjeev Gupta

+ 10:15-10:30 Tea Break

+ 10:30-11:45 Technical Session III: Social Media

Hate Speech Detection from Code-mixed Hindi-English Tweets Using Deep Learning Models
Satyajit Kamble and Aditya Joshi

Semi-Supervised Confidence Network aided Gated Attention based Recurrent Neural Network for Clickbait Detection
Amrith Rajagopal Setlur

User Perception of Code-Switching Dialog Systems
Anshul Bawa, Monojit Choudhury and Kalika Bali

Hierarchical Text Generation using an Outline
Mehdi Drissi, Olivia Watkins and Jugal Kalita

+ 11:45:13:00 NLP Software Contest Session-I

Monday, December 17, 2018 (continued)

+ 13:00-14:00 Lunch

+ 14:00-16:00 NLP Software Contest Session-II

+ 16:00-16:15 Tea

+ 16:15-17:30 Technical Session IV : Recent Trends

SMT vs NMT: A Comparison over Hindi and Bengali Simple Sentences
Sainik Kumar Mahata, Soumil Mandal, Dipankar Das and Sivaji Bandyopadhyay

Helping each Other: A Framework for Customer-to-Customer Suggestion Mining using a Semi-supervised Deep Neural Network
Hitesh Golchha, Deepak Gupta, Asif Ekbal and Pushpak Bhattacharyya

Towards Predicting Age of Acquisition of Words Using a Dictionary Network
Ditty Mathew, Girish Raguvir Jeyakumar, Rahul Kejriwal and Sutanu Chakraborti

Relation extraction between the clinical entities based on the shortest dependency path based LSTM
Dhanachandra Ningthoujam, Shweta Yadav, Pushpak Bhattacharyya and Asif Ekbal

+ 17:30-18:00 Valedictory Function

Abstractive Summarization Using Attentive Neural Techniques

Jacob Krantz
Gonzaga University
Spokane, WA
jkrantz@zagmail.gonzaga.edu

Jugal Kalita
University of Colorado, Colorado Springs
Colorado Springs, CO
jkalita@uccs.edu

Abstract

In a world of proliferating data, the ability to rapidly summarize text is growing in importance. Automatic summarization of text can be thought of as a sequence to sequence problem. Another area of natural language processing that solves a sequence to sequence problem is machine translation, which is rapidly evolving due to the development of attention-based encoder-decoder networks. This work applies these modern techniques to abstractive summarization. We perform analysis on various attention mechanisms for summarization with the goal of developing an approach and architecture aimed at improving the state of the art. In particular, we modify and optimize a translation model with self-attention for generating abstractive sentence summaries. The effectiveness of this base model along with attention variants is compared and analyzed in the context of standardized evaluation sets and test metrics. However, we show that these metrics are limited in their ability to effectively score abstractive summaries, and propose a new approach based on the intuition that an abstractive model requires an abstractive evaluation.

1 Introduction

The goal of summarization is to take a textual document and distill it into a more concise form while preserving the most important information and meaning. To this end, two approaches have historically been taken; extractive and abstractive. Extractive summarization selects the most important words of a given document and combines and rearranges them to form a final summarization (Nallapati et al., 2017). This approach is restricted to using words directly from the source document and so is unable to paraphrase. Abstractive algorithms generate a summary from an attempt to understand a document's meaning, allowing for paraphrasing much like a human may do. Abstractive approaches are more difficult to develop than extractive ones because an intermediate representation of knowledge is required. As such, dominant techniques of summarization have been extractive in nature, with wide-ranging solutions utilizing statistical, topic-based, graph-based, and machine learning approaches (Gambhir and Gupta, 2017). With the potential for generating more coherent and insightful summaries, abstractive approaches are gaining in popularity fueled by novel deep learning techniques (See et al., 2017). The abstractive summarization process includes converting words to their respective embeddings, computing a document representation, and generating output words. Neural networks have recently been shown to perform well for every step (Dong, 2018).

In deep learning models, attention allows a decoder to focus on different segments of an input while stepping through output regions. In the related sequence to sequence task of machine translation, attention was introduced to the existing encoder-decoder model (Bahdanau et al., 2014). This resulted in large improvements over past systems due to the ability to consider a larger window of context during the output generation. Progressing this further, Vaswani et al. (2017) showed that multi-headed self-attention can replace recurrence and convolutions entirely. As the areas of machine translation and abstractive summarization are related both structurally and semantically, the developments in machine translation may inform the direction of research in abstractive summarization. In this paper, we apply these advancements and

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 1–9

develop them further in pursuit of sentence summarization. In any attempt at summarization, the resulting text must be much more condensed than the original. In this task, all generated summaries are constrained to a fixed maximum length so that tested models must learn how to decide what information should be reproduced.

2 Related Work

Successful sentence summarization approaches have classically used statistical methods. TOPIARY (Zajic et al., 2004) detected salient topics that guided sentence compression while using linguistic transformations. MOSES, a statistical machine translation system, also performed well when directly used for summarization (Koehn et al., 2007). Attention mechanisms have been shown to improve the results of abstractive summarization. Rush et al. (2015) improved over classic statistical results by using a neural language model with a minimal contextual attention encoder. After the primary model training, an extractive tuning step was performed on an adjacent dataset. A related extension of this used a convolutional attentive encoder and experimented with replacing the decoder language model with RNN variants. LSTM cells and RNN-Elman both showed improved ROUGE scores (Chopra et al., 2016). An attentive encoder-decoder was also employed by Zeng et al. (2016) with one RNN architecture to re-weight another to improve context across the input sequence. Their decoder used attention with a copy mechanism that differentiated between out of vocabulary words based on their usage in the input. Nallapati et al. (2016) continued progress on encoder-decoder architectures by employing a bidirectional GRU-RNN encoder with a unidirectional GRU-RNN decoder. Imposing dynamic vocabulary restrictions also improved results while reducing the dimensionality of the softmax output layer. Pointer-generator networks encode with a bidirectional LSTM and decode with attention restriction. A coverage vector that limits the attention of words previously attended over is maintained (See et al., 2017).

Recently, summarization has made progress at the paragraph level due to reinforcement learning. A recurrent abstractive summarization model used teacher forcing and a similarity metric that compared the generated summary with the target summary (Paulus et al., 2017). The architecture contained a bi-directional LSTM with intra-attention. Actor-critic reinforcement learning was used by Li et al. (2018) to produce the highest scores for sentence summarization. One important consideration when optimizing purely on the test metric is that while overall recall is improved, higher ROUGE scores do not necessarily correlate with the readability of summaries.

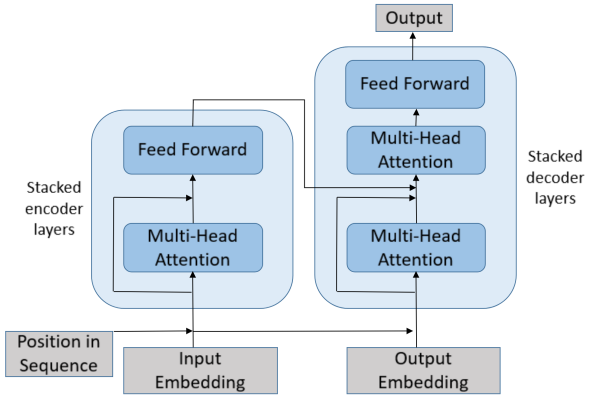


Figure 1: Transformer-based network architecture. The multi-headed attention mechanisms contain various recall options similar to and that expand upon Vaswani et al. (2017).

3 Models

Encoder-decoder architectures provide an adaptable structure for the development of systems that solve sequence to sequence problems. The encoder maps the input sequence to a latent vector representation. The decoder takes this representation, called the context vector, and generates the output sequence. The models and their variants that follow are structured as such. We select a base architecture that provides a strong foundation on which to analyze the effect of self-attention variants.

3.1 The Transformer

The Transformer architecture as proposed by Vaswani et al. (2017) is notable for performing state of the art Machine Translation, and is more efficient to train than past systems by orders of magnitude. This is made possible by replacing sequence aligned recurrence with parallel self-attention. The sequence order is preserved in the self-attention modules by including positional embeddings. Instead of incremental values, the positional embeddings are determined by position on a sinusoidal time series curve. Further, masking

of the decoder self-attention is performed, making the output of the next token dependent on that which has already been generated. Multi-headed self-attention is used in both the encoder and decoder. These mechanisms map a query vector to a key-value vector pair which results in an output vector. Tying together the encoder and decoder is a third multi-headed attention mechanism. The query comes from the self-attentive output of the decoder, and the keys and values from the self-attentive output of the encoder. In the work done by Vaswani et al. (2017), all attention heads used scaled dot-product attention, which is computationally efficient as multiple query, key, and value vectors can be implemented as a combined matrix. Scaled dot-product attention also defines the structure for the self-attention mechanisms we present below.

$$attention = softmax(\frac{QK^T}{\sqrt{d}})V \quad (1)$$

Many other attention mechanisms exist beyond the base dot-product attention. We analyze the performance of these mechanisms in the context of abstractive summarization. Changing the way the query, key, and value vectors interact allows an attention mechanism to learn different relationships between sequence elements.

Relative dot-product attention uses scaled dot-product attention, but instead of using absolute positional encodings, uses a relative positional encoding. These relative encodings learn to relate the elements of the query to both the elements of the keys and values (Gehring et al., 2017). The encodings can be distance-limited to a context window in the vector sequences.

Local attention divides the key-value vectors into localized blocks (Liu et al., 2018). Each query is strided over a corresponding block with a given filter size. Blocks can contain positions both prior to and following a given position, thereby not masking any element based on absolute position. Self-attention is performed over each block in isolation.

Local masked attention adds a mask to the blocks of local attention. Blocks in a future sequential position are masked from the query but all elements within a block remain visible to a given query position. Intuitively, masking future positions forces a mechanism to attend to current and past positions, which may be an important restriction of the attention distribution.

Local block masked attention masks both previous blocks and future blocks for a query position. Further, future positions within individual blocks are masked.

Dilated attention also divides the key-value vectors into blocks, but introduces a gap in between each block. Each query position is limited to a context window of a specified number of blocks both preceding and following the memory position.

Dilated masked attention performs the same operations as dilated attention and masks future memory positions within each block.

4 Evaluation

The standard test metric for automatic summary generation is ROUGE, or Recall-Oriented Understudy for Gisting Evaluation (Lin, 2004). Before the ROUGE metrics were introduced, human judges were used for summary evaluation. Human judges provide an ideal evaluation, but are impractical for regular use. ROUGE allows for automatic comparison of generated summaries to target summaries, where target summaries are human-generated. Limited-length recall is commonly reported using ROUGE-1, ROUGE-2, and ROUGE-L. ROUGE-1 and ROUGE-2 compare unigram and bigram overlap, respectively. This generalizes to ROUGE-N for n-gram overlap. ROUGE-L determines the longest common subsequence (LCS). Evaluation quality of summarization models can be directly compared to previous work because the same metrics were reported for past models by Rush et al. (2015), Zeng et al. (2016), Nallapati et al. (2016), Li et al. (2018), and others. These metrics allow for reasonably accurate comparison of summary generation models, but inherent problems exist. One critical limitation is that ROUGE does not consider equivalent paraphrasing or synonymous concepts. Since ROUGE works at the word level, meaning can only be captured and compared in a binary manner; either a word appears in the generated summary or it does not.

ROUGE 2.0 was proposed to alleviate this problem as well as remove the expectation that generated summaries need to be identical to the target summary (Ganesan, 2015). As pointed out by Rush et al. (2015), even the best human evaluator scored just 31.7 ROUGE-1 on the DUC2004

Target	*Endeavour astronauts join two segments of International Space Station.*
Gen1	Endeavour astronauts join two **sections** of International Space Station.
Gen2	Endeavour astronauts **remove** two segments of International Space Station.
Gen3	Endeavour astronauts join two segments of International Space Station.

Sentence	ROUGE-1	ROUGE-2	ROUGE-l	Cos-Sim	WMD	VERT
Gen1	88.89	75.00	88.89	0.979	0.418	94.77
Gen2	88.89	75.00	88.89	0.924	0.512	91.08
Gen3	100.00	100.00	100.00	1.000	0.000	100.00

Table 1: Highlighted differences between ROUGE and VERT scoring. Notice that an incorrect word replacement (*Gen2*) scores the same as a reasonable word replacement (*Gen1*) in ROUGE. VERT discounts the score of *Gen2* accordingly. *Gen3* is included to show the perfect scores for an identical summary.

dataset. This illustrates the idea that two summaries do not need to be the same in order for both to be of high quality. Thus, a more appropriate approach to summary comparison may be to evaluate the semantic similarity between the generated and target summaries instead of using isolated word counts. ROUGE 2.0 captures semantic similarity using a synonym dictionary while still evaluating n-grams and LCS. While this addresses the word-level shortcoming of the original ROUGE metrics, similarity is still fixed to a discrete list of acceptable alternatives, which does not fully capture phrase substitution. A further improvement could be to evaluate the semantic similarity between two entities on a continuous scale.

4.1 VERT Metric

To improve the quality of summary evaluation, we introduce the VERT metric[1], an evaluation tool that scores the quality of a generated hypothesis summary as compared to a reference target summary. VERT stands for Versatile Evaluation of Reduced Texts. VERT compares summaries on their underlying semantics rather than word count ratios. To calculate a VERT score for a summary pair, a similarity sub-score and dissimilarity subscore are calculated and functionally combined. Naturally, a higher similarity score and a lower dissimilarity score leads to a higher, better VERT score. The similarity sub-score considers the semantics of each summary taken at the document level. A sentence embedding vector is synthesized for both generated and target summaries, and the cosine similarity between these two vectors provides the similarity score. The sentence embeddings are generated using *InferSent*, an open-source neural encoder trained on natural language inference tasks (Conneau et al., 2017). *InferSent* was chosen because it has been shown to generalize well for use in various problems requiring sentence representations. The dissimilarity sub-score operates at the individual word level rather than at the sentence level. An aggregate Euclidean distance is calculated between the words of the generated summary and the words of the target summary. This is done using the word mover's distance (WMD) algorithm, a measure of how far document A must travel to match document B within a word vector space (Kusner et al., 2015). Stop words are discarded prior to the distance calculation as their effect on the distance between documents is negligible.

4.2 Sub-Score Motivations

A consideration would be to use just one of the two sub-scores as they are independent calculations. However, both the *InferSent* cosine similarity and WMD are made more robust by the presence of the other score. WMD is unaffected by word ordering, whereas the encoder of *InferSent* maintains sequential input. To illustrate, suppose the target sentence is *"go right and then left"* and the generated sentence switches the order, stating *"go left and then right."* WMD gives this a perfect distance of 0.0 but the *InferSent* similarity more accurately discounts the score by 4.3%. On the other hand, when longer summaries are compared, *InferSent* embeddings begin to lose the effect of individual words because the word embeddings are replaced with a singular embedding. This is less of a problem for WMD. Finally, the

[1]Our VERT implementation is made publicly available at: https://github.com/jacobkrantz/VertMetric

similarity sub-score uses GloVe embeddings[2] pre-trained on Common Crawl while the dissimilarity sub-score uses Word2Vec[3] trained on the Google News dataset. Using different word embeddings provides resistance to potential learned representation biases.

4.3 Formula Specification

The similarity sub-score is defined as $sim(s_1, s_2) = cos(encode(s_1), encode(s_2))$ and the dissimilarity sub-score is defined as $dis(s_1, s_2) = min(wmd(s_1, s_2), \alpha)$. The maximum dissimilarity value α is the default distance when all of the generated words are out of vocabulary. Without this default, summaries with no words to compare would have an infinite distance and too strongly influence VERT score averages. Resulting sub-score values range as such: $0.0 \leq sim(s_1, s_2) \in \mathbb{R} \leq 1.0$, and $0.0 \leq dis(s_1, s_2) \in \mathbb{R} \leq \alpha$. We seek to combine these scores such that the final VERT score can be treated as a percentage: $0.0 \leq VERT(s_1, s_2) \in \mathbb{R} \leq 1.0$. Further, $sim(s_1, s_2)$ and $dis(s_1, s_2)$ should be given equal weight in the final VERT score. To satisfy both criteria, we present the VERT equation:

$$VERT(s_1, s_2) = \frac{1}{2}(1 + (sim(s_1, s_2) - \frac{1}{\alpha} dis(s_1, s_2))) \quad (2)$$

where $\alpha = 5.0$. The dissimilarity is normalized by α and the outer linearity, as multiplied by $\frac{1}{2}$, shifts the range from $[-1.0, 1.0]$ to $[0.0, 1.0]$. For the choice of α, we observe an empirical distance ceiling of 5.0 in Table 2. Incorporating this ceiling gives both sub-scores equal precedence while removing the necessity of a nonlinearity, such as normalization by the hyperbolic tangent.

4.4 Hyperparameters and Baseline

The similarity sub-score uses a pre-trained *InferSent* encoder for reproducibility, and thus needs no hyperparameter adjustments. The dissimilarity requires just the hyperparameter α to specify the maximum threshold of WMD and can stay at the default value of 5.0. With the same value used to normalize the dissimilarity, VERT is straightforward to use with just this single hyperparameter.

[2] https://nlp.stanford.edu/projects/glove/

[3] https://code.google.com/archive/p/word2vec/

WMD	Summary Count
$0 \rightarrow 1$	74
$1 \rightarrow 2$	860
$2 \rightarrow 3$	2858
$3 \rightarrow 4$	2150
$4 \rightarrow 5$	58
5+	0

Table 2: WMD among human summaries on DUC2004. For each article, every human summary was held out as the target to compare the other human summaries to resulting in 6000 comparisons.

Metric	Pearson	P-Value
ROUGE-1	0.3039	0.0319
ROUGE-2	0.2577	0.0708
ROUGE-L	0.3071	0.0300
VERT	0.3681	0.0085

Table 3: Pearson correlation coefficient between automatic metrics and human evaluation of responsiveness.

To provide a scoring reference, we test each human summary of DUC2004 on VERT using the same holdout process as done in Table 2. The average similarity sub-score is 0.7488, the average dissimilarity sub-score is 2.7170, and combined the average VERT score is 0.6027.

4.5 Comparison to Human Evaluation

To evaluate the effectiveness of VERT, we calculate the correlation between VERT scores and scores given by human judges for generated sentence summaries. Using the relative dot-product attention model, 50 summaries are generated on the DUC2004 dataset and evaluated with the VERT metric by averaging the VERT scores between the four target summaries. We then conduct an experiment in which two human evaluators score the 50 generated summaries based on the DUC 2006 Responsiveness Assessment[4]. The primary consideration of responsiveness is the amount of information in the summary that relates to the original sentence. The evaluators score the level of responsiveness on a 5-point Likert scale, with 5 being the best possible. Table 3 shows that VERT correlates with human judgment of respon-

[4] https://duc.nist.gov/duc2007/responsiveness.assessment.instructions

siveness stronger than all three standard ROUGE metrics.

5 Experiments

5.1 Experiment Setup

The environment and evaluation of all models strictly follows the precedent set by Rush et al. (2015). For both training and testing, we extract sentence-summary pairs from news articles. The first sentence of each article is treated as the sentence to be summarized, while the headline of the article acts as the target summary.

5.2 Datasets

The training data comes from the Gigaword dataset, which is a collection of about 4 million news articles (Graff et al., 2003). It is necessary to discard certain article-headline pairs as some news articles open with a sentence that poorly relates to the headline, such as a question. Preprocessing tasks includes filtering, PTB tokenization, lowercasing, replacing digit characters with #, and replacing low-frequency words with *UNK*. Evaluation for hyperparameter tuning is performed on the DUC2003 dataset[5]. Testing is done on the DUC2004 dataset[6] where the summaries are capped at a length of 75 bytes. For both DUC2003 and DUC2004, each article has four target summaries to be compared against. For processing Gigaword, we used the same data provided by Rush et al. (2015), but both DUC datasets had to be preprocessed according to the tasks specified. Certain sentence-summary pairs within DUC 2004 poorly relate to each other due to the fact that the human-generated summaries used the context of the entire DUC article to decide on an adequate summary. Since this shortcoming is present across all models attempting sentence summarization on DUC, we made no effort to remove these difficult pairings from the test set.

5.3 Base Implementation

For the hyperparemeter specification, models used 8 attention heads and a dimension of 2048 for the dense feed forward layers. Cross entropy was used for the loss function, and optimization was performed with the Adam optimizer using a variable learning rate to encourage final convergence.

[5] https://duc.nist.gov/duc2003/tasks.html

[6] https://duc.nist.gov/duc2004/

Dataset	# Articles	Sent Len	Sum Len
Gigaword	3803957	31.4	8.3
DUC2003	624	32.7	11.2
DUC2004	500	31.3	11.7

Table 4: Comparison of general dataset details. Sentence and summary lengths are reported as the average word count. Gigaword has noticeably shorter target summaries than either DUC dataset. To counteract the models generating too short of summaries, we augment the beam search decoding probabilities to encourage longer summaries.

Training required approximately 25 epochs. A promising feature of using an attention-based architecture is that the models used here are capable of being trained in approximately 4 hours on a single GPU, whereas recent state of the art recurrent summarization models have been mentioned to take 4 days (Rush et al., 2015). We implemented these models using the Tensor2Tensor[7] library backed by TensorFlow. A strong local minimum exists when training, which closely relates to extracting the first n words of the input text up to 75 bytes. Such a trivial approach produces relatively high ROUGE scores simply due to the natural similarity between target summaries and input sentences. Diversity of attention can be encouraged by varying the learning rate and modifying the attention mechanism itself. For the decoding step, beam search is used with a beam size of 8. This results in ROUGE scores that are higher than a more simple greedy inference. Decoding to a fixed length of 75 bytes does not align easily with word-level decoding, so for the implementation we approximate the cutoff by limiting the summary sequence to 14 words.

6 Results

6.1 Attention Comparisons

For each of the attention mechanisms described above, we performed a full scale analysis of their performance by training each model on the Gigaword dataset and evaluating on DUC2004. For each experiment, the foundational architecture was held constant. We modified both the encoder self-attention and decoder self-attention to perform as specified by the given attention mechanism. In Table 5, the model that used scaled

[7] https://github.com/tensorflow/tensor2tensor

Mechanism	RG-1	RG-2	RG-L	VERT-S	VERT-D	VERT
s-dot-prod	25.72	8.51	23.08	0.73523	2.76307	59.13
rel-s-dot-prod	**27.05**	**9.54**	**24.44**	**0.73876**	**2.73907**	**59.55**
local	1.93	0.00	1.93	0.02084	5.00000	1.04
local-mask	25.72	8.54	23.30	0.73361	2.77857	58.89
local-blk-mask	14.13	2.75	12.63	0.67226	3.18881	51.73
dilated	0.01	0.00	0.01	0.09509	3.66543	18.10
dilated-mask	19.06	5.23	17.45	0.68682	3.04922	53.85

Table 5: Comparison of attention mechanisms using DUC2004. RG represents ROUGE-Recall, VERT-S is the *InferSent* cosine similarity sub-score, and VERT-D is the average WMD sub-score.

Model	RG-1	RG-2	RG-L	VERT
TOPIARY (Zajic et al., 2004)	25.12	6.46	20.12	-
ABS (Rush et al., 2015)	26.55	7.06	22.05	58.49
RAS-LSTM (Chopra et al., 2016)	27.41	7.69	23.06	-
MOSES+ (Koehn et al., 2007)	26.50	8.13	22.85	-
RAS-Elman (Chopra et al., 2016)	28.97	8.26	24.06	-
ABS+ (Rush et al., 2015)	28.18	8.49	23.81	59.05
RA-C-LSTM (Zeng et al., 2016)	29.89	9.37	25.93	-
words-lvt5k-1sen (Nallapati et al., 2016)	28.61	9.42	25.24	-
S-ATT-REL (ours)	27.05	9.54	24.44	59.55
AC-ABS (Li et al., 2018)	32.03	10.99	27.86	59.67

Table 6: ROUGE-recall scores of compared models on DUC2004. Sorted by ROUGE-2 score. ABS, ABS+, and AC-ABS VERT scores were calculated using summaries provided by their respective authors.

dot-product attention acted as the baseline (s-dot-prod). The highest performing mechanism was relative scaled dot-product attention, showing that relative positional encodings can be more insightful than absolute encodings. This demonstrates that token generation may rely more heavily on the relationships between surrounding words than relationships at a global sequential level. Local masked attention attained marginally higher ROUGE-2 and ROUGE-L scores than scaled dot-product attention. However, scaled dot-product attention scored noticeably higher with VERT, primarily due to the similarity sub-score. This suggests the scaled dot-product model is better than the local-mask model when considering the summary semantics across an entire sequence. Both local and dilated attention mechanisms repeated the same words regardless of input sentence; both masked counterparts did not have this problem.

We found a high dependence on batch size during the training process. Models would not converge when batch sizes were at or below 2000 tokens per batch. The batch size used to train the above models was 8192 tokens. Dilated attention and dilated-mask attention models were trained at lower batch sizes due to higher memory requirements. This may have negatively effected results.

6.2 Model Comparisons

We compare our best model with past work by comparing published ROUGE scores. Slight variances may be present in the reported metrics due to potential differences in data preprocessing routines. In Table 6, we compare our best model with that of published results. The relative dot-product self-attention model (S-ATT-REL) beats all ROUGE scores of ABS, but has a lower ROUGE-1 when ABS is tuned with an extractive routine on DUC2003 (ABS+). S-ATT-REL is comparable to but lower than certain models when it comes to ROUGE-1 scores. However, over the longer subsequence comparisons of ROUGE-2 and ROUGE-L, S-ATT-REL performs very well. This can be attributed to the ability of self-attention mechanisms to retain a strong memory over past elements of the input and decoded sequences. Only the actor-critic method (AC-ABS) beats S-ATT-REL in all tested categories.

6.3 Qualitative Discussion

The summaries generated by our best model are strongly abstractive, illustrated by Example *S(1)* in Figure 2. Example *S(2)* showcases the ability to utilize long range recall. From the appositive phrase, the model determined that Hariri was the prime minister of Lebanon and adjusted the morphology of the country for succinctness. The model also determined Hariri was resigning based on the words "bowing out". Occasionally, attention heads are misdirected and attend to words or phrases that do not contain the primary meaning. This occurred in Example *S3* with was incorrectly modified by the inclusion of "not". The generated summaries exhibit information beyond what was directly in the input sentence; Example *S5* correctly identifies Premier Romano as Italian which greatly improves the informedness of the summary. A primary strength of the self-attentive model is incorporating abstract information from all segments of the input sentence. This is suggested in the long subsequence ROUGE scores above, and seen clearly in qualitative analysis.

An assessment of linguistic quality[8] was performed alongside the DUC Responsiveness Assessment. This followed the same procedure detailed in Section 4.5. Questions pertained to grammaticality, non-redundancy, referential clarity, and structure and coherence. Grammaticality scored 4.48, non-redundancy scored 4.95, referential clarity scored 4.7, and structure and coherence scored 4.53. All scores averaged between "Good" and "Very Good". Non-redundancy is nearly perfect, likely because the summaries are too short for redundancy to be of issue. The referential clarity scored high as well, which can be associated with the performance of the self-attention over the words already decoded.

S(1): exxon corp. and mobil corp. have held discussions about combining their business operations , a person involved in the talks said wednesday . **Target:** exxon corp. and mobil corp. may combine business operations **S-ATT-REL:** exxon and mobil discuss merger
S(2): prime minister rafik hariri , the business tycoon who launched lebanon 's multibillion dollar reconstruction from the devastation of civil war , said monday he was bowing out as premier following a dispute with the new president . **Target:** prime minister hariri , claiming constitution violation , bows out **S-ATT-REL:** lebanese prime minister resigns after dispute with new president
S(3): organizers of december 's asian games have dismissed press reports that a sports complex would not be completed on time , saying preparations are well in hand , a local newspaper said friday . **Target:** bangkok says sports complex will be completed in time for asian games **S-ATT-REL:** asian games organizers say sports complex will not be completed on time
S(4): a struggle for control of the house is under way , with rep. robert livingston conducting a telephone campaign that could lead to him running against newt gingrich as speaker . **Target:** election of gingrich as house speaker in doubt as small group opposes him **S-ATT-REL:** house speaker 's phone campaign could lead to gingrich
S(5): premier romano prodi battled tuesday for any votes freed up from a split in a far-left party , but said he will resign if he loses a confidence vote expected later this week . **Target:** italian premier to resign if he loses pending confidence vote **S-ATT-REL:** italy 's prodi says he will resign if he loses confidence vote

Figure 2: Examples of generated summaries by the relative dot-product self-attention model.

7 Conclusion

The effect of modern attention mechanisms as applied to sentence summarization has been tested and analyzed. We have shown that a self-attentive encoder-decoder can perform the sentence summarization task without the use of recurrence or convolutions, which are the primary mechanisms in state of the art summarization approaches today. An inherent limitation of these existing systems is the computational cost of training associated with recurrence. The models presented can be trained on the full Gigaword dataset in just 4 hours on a single GPU. Our relative dot-product self-attention model generated the highest quality summaries among our tested models and displayed the ability of abstracting and reducing complex dependencies. We also have shown that n-gram evaluation using ROUGE metrics falls short in judging the quality of abstractive summaries. The VERT metric has been proposed as an alternative to evaluate future automatic summarization based on the premise that an abstractive summary should be judged in an abstractive manner.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1659788 at the University of Colorado, Colorado Springs REU site.

[8] https://duc.nist.gov/duc2007/quality-questions.txt

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Yue Dong. 2018. A survey on neural network-based summarization methods. *arXiv preprint arXiv:1804.04589*.

Mahak Gambhir and Vishal Gupta. 2017. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1):1–66.

Kavita Ganesan. 2015. Rouge 2.0: Updated and improved measures for evaluation of summarization tasks. *arXiv preprint arXiv:1803.01937*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pages 1243–1252.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*, 4:1.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966.

Piji Li, Lidong Bing, and Wai Lam. 2018. Actor-critic based training framework for abstractive summarization. *arXiv preprint arXiv:1803.11070*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Proceedings of the ACL Workshop: Text Summarization Branches Out*.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 280–290.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

David Zajic, Bonnie Dorr, and Richard Schwartz. 2004. Bbn/umd at duc-2004: Topiary. In *Proceedings of the HLT-NAACL 2004 Document Understanding Workshop, Boston*, pages 112–119.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efficient summarization with read-again and copy mechanism. *arXiv preprint arXiv:1611.03382*.

Challenges and Issues in Developing an Annotated Corpus and HMM POS Tagger for Khasi

Medari Janai Tham
Computer Science & Engineering and Information Technology
Assam Don Bosco University
medaritham16@gmail.com

Abstract

An attempt has been made to annotate a Khasi corpus with Part-of-Speech (POS) tags, using the Bureau of Indian Standards (BIS) POS tagset prepared by the POS Tag Standardization Committee of the Department of Information Technology (DIT), New Delhi, India for annotating Indian language corpora. This is the first initiative taken for Khasi- an understudied and under-resourced language, in developing an annotated corpus and POS tagger essential for language technology. This article highlights the challenges and issues that surfaced during annotation, and the decisions that were taken when tagging features characteristic of Khasi that are absent from mainstream Indian languages. A Hidden Markov Model (HMM) POS tagger is then constructed, taking into consideration the information provided by the morphological features of the Khasi language. The results of training and testing the Khasi HMM POS tagger are compared with the results of a Khasi baseline tagger, and a Khasi tagger constructed using Natural Language Toolkit (NLTK).

1 Introduction

Construction of resources is necessary for natural language processing and this article describes the process initiated in the development of an annotated corpus and POS tagger for Khasi, which are basic resources required for natural language applications such as parsing, information retrieval, question and answering, etc.

Standard guidelines in annotating text corpora are essential when an attempt is made to annotate a corpus from scratch as is the case with Khasi. The benefits of annotating the corpus using the prescribed standard such as Bureau of Indian Standards (BIS) (Chaudhary et al., 2010) for Indian languages will facilitate the corpus in inter linguistic analysis and study. The annotated Khasi corpus has been constructed from a collection of Khasi literature of prose and fiction genre and it comprises of 3,984 sentences which include 86,087 tokens out of which 75,736 are tokens excluding punctuation and 5,313 word types. The applied BIS tagset for Khasi is given in Table 1 and Table 2. The questions and issues that emerged when annotating the corpus and their proposed suggestions are discussed in section 6. The construction and analysis of the Khasi HMM POS tagger and the comparison of its results with the Khasi baseline tagger and the Khasi NLTK tagger are given in section 7.

2 Related Work

POS tagging is the process of automatically assigning a part of speech to each word present in a corpus. These part of speech tags are assigned from a specific tagset applicable to the language. Current POS tagging accuracy is about 96%-97% for languages such as English, French, etc. (Güngör, 2010). Approaches to tagging algorithms are either rule-based taggers or stochastic taggers. The most widely used tagger in rule-based tagging is the Transformation Based Learning (Brill, 1995) often called the "Brill tagger". This approach also uses machine learning to learn the rules form the data and achieved 96.6% accuracy when trained and tested on the WSJ corpus. On the other hand, stochastic taggers utilize the availability of lexicons and corpora and one such learning approach is the Hidden Markov Model (HMM) which has obtained high accuracies in POS tagging. For example, the most available tagger and highly accurate is the TnT tagger (Brants, 2000). Its influence comes from its sensitive dealing with unknown words and smoothing. Another HMM tagger is the HunPos trigram tagger (Halácsy et al., 2007) which unlike TnT, provided mechanisms where a language morphological features can be tweaked into the

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 10–19

tagger and achieved 98.24% accuracy for Hungarian when compared to TnT's 97.42% on the same corpus. However, the TnT tagger remains one of best performing taggers across different languages (Plank et al., 2016).

According to the 2001 Indian census, the language families present in India are Indo-Aryan, Dravidian, Austro-Asiatic, Tibeto-Burmese and Semito–Hamitic. Among these language families, Indo-Aryan and Dravidian are the two major family groups of India comprising approximately 97% of India's population. A recurring pattern with stochastic POS taggers developed for Indian languages, is that they have to content with small size training data and language specific tagsets. Reported tagging accuracies for Indian languages range from 69%-96% and some of the POS taggers developed for Indo-Aryan (Hindi), Tibeto Burman (Manipuri and Kokborok) and Dravidian (Tamil) families are as follows.

Apart from English, Hindi is the official language of India. It is a morphologically rich language, and one POS tagger (Singh et al., 2006) developed for Hindi has taken advantage of this feature to compensate the lack of annotated corpora by utilizing extensive morphological analysis along with a high-coverage lexicon and decision tree based learning algorithm where the size of the corpus used is 15,562 words, and achieved POS tagging accuracy of 93.45%. Another POS tagger (Shrivastava and Bhattacharyya, 2008) for Hindi that does away with the need of a morphological analyzer and structured lexicon, uses the HMM approach where a list of all possible suffixes in Hindi is employed to perform stemming on a corpus of 66,900 words and achieved 93.12% accuracy. On the other hand a rule based Hindi POS tagger (Garg et al., 2012) reported an accuracy of 87.55%.

In the absence of tagged corpora, a morphologically driven POS Tagger for Manipuri (Singh and Bandyopadhyay, 2008) achieved 69% accuracy tested on 3,784 sentences. A Manipuri POS tagger (Singh et al., 2008) using condition random field and support vector machine trained on 39,449 tokens and tested on 8,672 tokens reported 72.04% and 74.38% accuracies respectively. Another condition random field Manipuri POS tagger (Nongmeikapam and Bandyopadhyay, 2012) used for transliterating from Bengali script to the Meitei Mayek script achieved a precision of 74.31%, a recall of 80.20% and an F-measure of 77.14%. POS taggers developed for Kokborok, another resource constrained language (Patra et al., 2012), include rule based tagger with 69% accuracy and stochastic taggers using condition random field and support vector machine with 81.67% and 84.46% accuracies respectively.

POS taggers for Tamil include rule based (Selvam and Natarajan, 2009) with 85.56% accuracy, and a morpheme based language model (Pandian and Geetha, 2008) involving 35 tags and a test set of 43,678 words with 95.92% accuracy.

3 Concise Overview of BIS

The BIS standard has been prepared to work for languages even beyond Indo-Aryan and Dravidian families and the guidelines have been formulated taking into account existing tagsets designed under various projects such as the Indian Language Machine Translation (ILMT) POS tagset (Bharati et al., 2006), Microsoft Research India Indian Language POS (Baskaran et al., 2008) and others. Taking into consideration the existence of various language families in India, the tagset has been designed to be all accommodating. The annotation follows a layered approach where the linguistics features can be incorporated in layers such as morphology in one layer, part of speech in another layer, syntactic analysis in another layer and the others in different layers, and within each layer there is a hierarchy of categories. Extensibility is a key feature of the tagset where a top category or a sub category can be added to the existing hierarchy if the language under question requires one. On the other hand, a tag may not be utilized if it is not required even if it exist in the BIS tagset. The POS tagging has to be carried out on text that have been pre-processed, where each token in the corpus is a single lexical item and any morphological analysis required should have been processed by a morphological analyzer. In total, the tagset has 11 top level categories with very few categories having two levels of subtypes, reflecting the coarse nature of the tagset.

4 Brief Introduction to Khasi Language

Khasi is classified under the Mon-Khmer branch of the Austro-Asiatic language family (Diffloth, 2018). It is the associate official language of the state of Meghalaya, India and according to the

2011 Indian census there are approximately 1.4 million speakers in Meghalaya and Assam, placing it less than 1 percent of India's population. Khasi is an analytic and non-inflectional language exhibiting derivational morphology which contributes to the partial agglutinative behavior of the language (Nagaraja, 2000).

Khasi is written in the Latin script comprising of 23 letters where the letters *c*, *f*, *q*, *v*, *x*, *z* have been removed with the addition of the diacritic letters *ï* and *ñ* and the diagraph *ng* which is adopted as a single letter.[1]

5 Khasi Corpus Construction

A corpus is designed to represent a particular natural language or language variety by virtue of the range of text included and the sampling from each text used in collecting the data contained in the corpus (Xiao, 2010). Due to the unavailability of any corpus in Khasi, the required corpus has to be built from scratch which consumes time and effort. The data collected for the current corpus are samples from the prose and fiction genre of Khasi literature that are prescribed for studies in higher secondary, graduation, and post-graduation. The selection of Khasi literature is compelled by the fact that though newspapers are easily available online, they are not accepted by language experts as a representation of the language because of the lack of consensus on how the language should be written in terms of its grammar and orthography. On the other hand, it is also observed that in most instances, the written literature does not conform to any single standard when it comes to orthography even within text written by the same author. To cite a few examples the preposition *ïa* (to) is written as *ïa* or *ia*, where the word is written with the letter *i* with diaeresis or without it. Other examples are the words *duai* (pray) where it is also written as *duwai*, *mynmied* (night) which it is also written as *mynmiet*, etc. Another category of a nominal that do not follow a uniform orthography are doublets. These are two nouns that occur together having the same semantics and are often used more for their stylistic value. A few examples are *ki-mrad ki-mreng* (animals), *ki khun-ki kti* (children) , *u-kñi-u-kpa* (ancestor) where the hyphen (-) is used according to the author's style.

[1] http://www.ciil-lisindia.net/Khasi/Khasi.html

In analyzing natural language in digital format it is necessary that the characters, words and sentences are clearly identified before any natural language processing task can be carried out. This process of dividing a text document into words and sentences is called text segmentation. Khasi utilizes the Latin script for writing and like English the whitespace is used to marked word boundaries. The data for analysis is pre-processed manually where each word is separated by a space and each sentence is marked with an end of sentence marker such as a period (.), a question mark (?) or an exclamation mark (!). Thus the words identified are also called tokens and these tokens include punctuations. This implies that the punctuations are not attached with a word but are delimited with a whitespace. The only exception is the use of apostrophes (') and the hyphens (-). The apostrophe is used to marked contractions such as *bar'bor* (everytime), and the hyphen to form compound words such as *Khasi-Khara* (the Khasis); the reduplicated forms often used with adverbs such as *khah-khah* (regularly) where these punctuations are also part of the tokens. The corpus is then manually tagged using the BIS tagset shown in Table 1 and Table 2.

6 Annotating Khasi using BIS tagset

This section discusses the challenges and the issues faced when tagging Khasi and the decisions that were taken on encountering features prevalent in the language. The grammatical characteristics of the language taken into consideration are with references to the works of various contributors on the Khasi language (Rabel, 1961; Bars, 1973; Henderson, 1976; Nagaraja, 1985; Jyrwa, 1989; Roberts, 2005; War, 2011)

6.1 Personal Pronouns

The structure of personal pronouns in Khasi is simple except in the case of third person singular and plural forms. Apart from their basic functionality, third person singular and plural personal pronouns such as: /i/ 'singular, neutral', /u/ 'singular, masculine' /ka/ 'singular, feminine' and /ki/ 'common, plural' also function as number and/or gender markers. The personal pronoun *i* when used, indicates reverence or refers to diminutive objects. They are also described as articles, determiners, gender indicators and pronominal markers. It is mandatory that every noun in Khasi is preceded by pronominal markers

Categories						
Sl. No	Top Level	Subtype (Level 1)	Subtype (Level 2)	Label	Annotation Convention	Example(s)
1	Noun			N	N	
1.1		Common		NN	N_NN	jingsuk 'peace' ksew 'dog'
1.2		Proper		NNP	N_NNP	Melam, Shillong
1.3		Nloc		NST	N_NST	sha-lor LOC-top 'on top'
2	Pronoun			PR	PR	
2.1		Personal		PRP	PR_PRP	nga 1S 'I'
2.1.1			Pronominal	PRP_M	PR_PRP_M	ka kot PM book 'a/the book'
2.1.2			Auxiliary	AUX	PR_PRP_AUX	nga-n 1S-FUT 'I will'
2.2		Reflexive		PRF	PR_PRF	lade 'self'
2.3		Relative		PRL	PR_PRL	u-ba 3SM-that 'he that'
2.4		Wh-word		PRQ	PR_PRQ	u-ei 3SM-who 'who'
2.5		Indefinite		PRI	PR_PRI	ka-no ka-no 3SF-whoever 3SF-whoever 'whoever'
3	Demonstrative			DM	DM	
3.1		Deictic		DMD	DM_DMD	ka-ta 3SF-out of sight 'that'
4	Verb			V	V	
4.1		Main		VM	V_VM	bam 'eat'
4.2		Auxiliary		VAUX	V_VAUX	lah 'can'
4.2.1			Infinitive	VINF	V_VAUX_VINF	ban 'to'
5	Adjective			JJ	JJ	bakhraw 'great'

Table 1: Khasi BIS Tagset

(PM) which are third person personal pronouns. Exceptions where the pronominal marker is dropped are in vocative sentences, optionally in locative phrases where inanimate nouns are used,

Categories						
Sl. No	Top Level	Subtype (Level 1)	Subtype (Level 2)	Label	Annotation Convention	Example(s)
6	Adverb			RB	RB	suki-suki ‘slowly-slowly’
7	Conjunction			CC	CC	
7.1		Coordinating		CCD	CC_CCD	bad ‘and’
7.2		Subordinating		CCS	CC_CCS	namar ‘because’
8	Particles			RP	RP	
8.1		Default		RPD	RP_RPD	noh PRT
8.2		Classifier		CL	RP_CL	tylli
8.3		Interjection		INJ	RP_INJ	wa, ada
8.4		Intensifier		INTF	RP_INTF	shuh, eh
8.5		Negation		NEG	RP_NEG	ki-m 3PL-will not ‘they will not’
8.6		Possessive		POS	RP_POS	la POS
9	Quantifiers			QT	QT	
9.1		General		QTF	QT_QTF	shi ‘one’
9.2		Cardinals		QTC	QT_QTC	wei ‘one’
9.3		Ordinals		QTO	QT_QTO	banyngkong ‘first’
10	Residuals			RD	RD	
10.1		Foreign		RDF	RD_RDF	a word not written in Khasi
10.2		Symbols		SYM	RD_SYM	#, $
10.3		Punctuation		PUNC	RD_PUNC	; ,
10.4		Unknown		UNK	RD_UNK	
10.5		Echowords		ECH	RD_ECH	lyngaiň
11		Preposition		IN	IN	na ‘from’

Table 2: Khasi BIS Tagset cont...

and when nouns immediately follow a verb- they blend with the verb and cease to be nouns (Jyrwa, 1989). Another functionality of these pronominal markers is their occurrence before a verb (also called subject enclitic (Jyrwa, 1989)) indicating subject verb agreement and highlighted in bold in the example below.

ka Iba **ka** ai ka kot
PM Iba PM give PM book
‘Iba gave the book’

Ideally, tagging them as pronominal markers will be appropriate in highlighting the fact that they stand in agreement with the head noun, but there are instances where their occurrences can also be

quite far from the head noun which is not feasible for machine learning purposes in disambiguating them from personal pronouns. Therefore, the new tag PR_PRP_M representing a pronominal marker is applied only to pronouns occurring before a noun only and not for subject enclitic. The existing personal pronoun tag PR_PRP has been maintained for subject enclitic.

Another problem is the personal pronouns attached with the suffix -n or -m such as *ngan* (I will), *ngam* (I will not), etc. For instance *ngan* indicates tense equivalent to English (will) and (shall) and *ngam* indicates tense and negation (will not) and (shall not). As per BIS guidelines, any morphological analysis required must have been carried out before tagging, such that each token is a lexical item and requires no further processing. If morphological analysis is applied to these pronouns, we now have *ngan* (I will) mapping to *nga* 'first person, singular' and *yn* (will) an auxiliary verb. *Ngam* on the other hand will have two mappings-- a) (I will not) mapping to 'first person, singular' and *ym* (will not) an auxiliary verb and b) *ngam* (drown) which is a verb. Since this analysis is applicable to a finite number of words, a morphological analyzer is not employed, and specifically when these words function as pronouns in the corpus, they are given a newly created tag PR_PRP_AUX which is a sub-type of the personal pronoun category. It may be mentioned that these words do not function as pronominal markers.

6.2 Multi-functionality of la

The word *la* in Khasi can function as a past tense marker or an auxiliary verb or particle or a possessive particle or a subordinating conjunction. When *la* functions as an auxiliary verb or a past tense marker, it has been tagged as an auxiliary verb V_VAUX because their occurrences in sentences are syntactically similar. The BIS tagset has provisions for subordinating conjunction and particle but not for possessive marker. While tagging, the tags applied for subordinating conjunction and particle are CC_CCS and RP_RPD respectively. Again, keeping in mind BIS extensibility feature, a new sub-type of the particle category RP_POS is created to accommodate *la* functioning as a possessive particle.

6.3 Tagging of Adverbs

The BIS tagset specifies that only manner adverbs should be tagged as RB such as *ïaid suki suki* (walk slowly). It appears that no BIS tag is appropriate for adverbs such as *ruh* (also), *ju* (in the habit of, used to), etc. In order to maintain minimalism of new tags in the BIS tagset, in the present corpus any occurrences of such words are still tagged as RB.

6.4 Absence of Prepositions in BIS Tagset

BIS has incorporated postpositions with the tag PSP which is a prevalent feature in the Indo-Aryan and Dravidian families but absent in Khasi. Khasi utilizes prepositions and in order to accommodate them, a new top level category is constructed with tag IN.

6.5 Nouns of Location Space and Time (Nloc)

The BIS tagset clearly states that only a finite number of nouns of location, space and time that can also function as postpositions are tagged as N_NST. The question was, whether this category is also applicable to Khasi or not. From the literature and the data in the corpus, it came to attention that a certain group of words in Khasi can function as a noun or a preposition or as an adverb depicting the behavior mentioned in the BIS specification. These are compound words comprising of a preposition (ha/na/sha) and a bound or a free element. These words are also referred as prepositional adverbials such as *halor* (on top), *sharum* (downwards/south), etc. The conclusion that was brought forward in the BIS tagset specification is to facilitate machine learning and simultaneously avoid confusion in annotation- therefore in the present corpus they have been uniformly tagged as N_NST irrespective of their function.

6.6 Tagging Compound Words and Imitative

Compounds in Khasi are primarily formed when a space or a hyphen separates the elements of the compound word, or they are collocated. For example, *khia thew* (graceful), *bai-sngi* (wage) and *metbneng* (planets). The compound word that is written as a single word or where the elements of the compound are separated by hyphens is tagged by taking its grammatical function in the sentence.

7 Applying the Hidden Markov Model for POS tagging

7.1 POS Tagging

During POS tagging, each word in the corpus is automatically tagged with its part of speech. Therefore, given an input string of words and a tagset the output of a POS tagger should be the best possible tag for each word. For example, using the BIS tags for Khasi from Table 1 and Table 2, a sentence in Khasi is tagged as follows.

```
Tiap\RB tang\RB shu\RB poi\V_VM
ha\IN bri\N_NN ,\RD_PUNC u\PR_PRP_M
slap\N_NN u\PR_PRP sdang\V_VM
hap\V_VM .\RD_PUNC
```

'Immediately when he reached the field the rain started falling'

7.2 Hidden Markov Model Approach

Given a tagset, in this instance the BIS tagset in Table 1 and Table 2, and a sentence of n words W= $w_1,w_2,...w_n$, the POS tagger has to find the sequence T= t_1, $t_2...t_n$, where T is a set of tags from the tagset that satisfies the following equation.

$$argmax_T \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1} \dots t_{i-k}) \quad (1)$$

In other words the best possible tag sequence is a sequence that maximize the lexical P(W|T) and transition P(T) probabilities. Since the tags are hidden and only the words are observed we have a hidden Markov model where states represent the tags and the outputs are the observed words. In the lines of Brants (2000) TnT tagger, a second order Markov model is used where k=2 in equation 1 and adding tags t_{-1}, t_0, and t_{n+1} for beginning of sentence and end of sentence markers. Equation 1 is now calculated as follows.

$$argmax_T(\prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1},t_{i-2}))P(t_{n+1}|t_n) \quad (2)$$

Using an annotated corpus, the probabilities in equation 2 are estimated using the maximum likelihood estimation.

$$P(w_i|t_i) = \frac{f(w_i,t_i)}{f(t_i,)} \quad (3)$$

$$P(t_i|t_{i-2},t_{i-1}) = \frac{f(t_{i-2},t_{i-1},t_i)}{f(t_{i-2},t_{i-1})} \quad (4)$$

where f(w,t) is the number of occurrences of words w with tag t and f($t_1,t_2,...t_m$) is the number of occurrences of the tag sequence $t_1,t_2,...t_m$.

We can compute equation 2 for each possible tag sequence of length n and then take the sequence with the highest probability. However the complexity of this algorithm is exponential to the number of words. An efficient algorithm operating in linear time is the Viterbi (Rabiner, 1989) algorithm which is used here to determine the optimal sub paths rather than keeping track of all paths during execution. The trigram tagger given in equation 2 has one problem and that is data sparsity. Any trigram instance in the test set may not have occurred in the training set implying that equation 4 will give zero probability and in turn give rise to zero probability tag sequences. Considering **N** as the total number of tokens in the training corpus, from equation 4 the maximum likelihood estimation can be calculated as follows

$$Trigram\ \hat{P}(t_i|t_{i-2},t_{i-1}) = \frac{f(t_{i-2},t_{i-1},t_i)}{f(t_{i-2},t_{i-1})} \quad (5)$$

$$Bigram\ \hat{P}(t_i|t_{i-1}) = \frac{f(t_{i-1},t_i)}{f(,t_{i-1})} \quad (6)$$

$$Unigram\ \hat{P}(t_i) = \frac{f(t_i)}{N} \quad (7)$$

As suggested in Jurafsky and Martin (2009), linear interpolation can be used and we now estimate the probability as

$$P(t_i|t_{i-2},t_{i-1}) = \lambda_3\hat{P}(t_i|t_{i-2},t_{i-1}) + \lambda_2\hat{P}(t_i|t_{i-1}) + \lambda_1\hat{P}(t_i) \quad (8)$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$

In order to approximate the value of λ Brants (2000) version of deleted interpolation is used for setting the λ's.

7.3 Using Morphology in Handling Unknown Words

As mentioned in section 4, Khasi exhibits derivational morphology in the form of agglutination by adding affixes to word base to derive other words. These affixes can be easily separated from the root and the focus here are on the prefixes attached to Khasi nouns and verbs. Khasi words reveal that words with prefixes such as jing-, nong- and maw- always map to common nouns (N_NN). Words with prefixes such as pyn- and ïa- excluding the preposition ïa, always map to verbs (V_VM). It may be noted that *pynban* (cause to press) which is a verb can also function as an adverb (nonetheless).

In the training and test data, the words having prefixes jing- are mapped to pseudo-word _JING_, nong- to pseudo-word _NONG_, maw- to pseudo-word _MAW_, pyn- are mapped to pseudo-word _PYN_ and ïa- excluding preposition ïa, are mapped to pseudo-word _IA_.

This mapping is carried out for data in the training set and in the test set, to estimate the probabilities of unknown words having these prefixes.

In order to handle unknown words not having the above mentioned prefixes, low frequency words in the training data are mapped to pseudo-word _UNK_. Similarly, words in the test set that were unseen in the training data are also mapped to pseudo-word _UNK_. Since the corpus size is relatively small, it is observed that words occurring only once in the training set account to 49.1% of the training data. Therefore low frequency is taken to be less than or equal to a selected value γ and in this tagger γ=1.

After the mappings are done, the HMM parameters are evaluated as mentioned earlier where the pseudo-words _JING_, _PYN_, _NONG_, _IA_ and _UNK_ are treated like regular words. This mapping is carried out to ensure that the probability of $P(w_i|t_i)$ is never zero.

7.4 Testing and Evaluation

The corpus has been divided into training set and test set. The training set consists of 3,984 sentences comprising of 86,087 tokens and 5,313 word types. The test set consists of 402 sentences which include 8,565 tokens and 1,110 word types. The test set is a sample from a book not included in the training set.

The data has been tested using a baseline tagger, an NLTK tagger, and the HMM POS Tagger and the results are shown in Table 3. As proposed by Jurafsky and Martin (2009), the baseline tagger tags the words in the test data with their most frequent tag obtained from the training data.

NLTK (Bird et al., 2009) also provides taggers such as the trigram tagger, bigram tagger, default tagger and regular expression tagger. Taking into account the morphological features of Khasi mentioned in section 7.3, an NLTK tagger for Khasi was constructed where an NLTK trigram tagger backs off to a bigram tagger, the bigram tagger backs off to a unigram tagger and the unigram tagger backs off to a Khasi regular expression tagger. The Khasi regular expression tagger tags words with prefixes jing-, nong-, and maw- as common nouns (N_NN), words with prefixes pyn- and ïa- as verbs (V_VM) and defaults to the most common tag which is the common noun (N_NN). Words having frequency less than or equal to 1 in the training data and unseen words in the test data are also mapped to the pseudo-word _UNK_ to handle unknown words. However, the words having the above mentioned prefixes are not mapped to _UNK_ since the tagger eventually backs off to the Khasi regular expression tagger. Additionally, Table 3 also highlights results of the NLTK bigram tagger which backs off to a unigram tagger and an NLTK trigram tagger which backs off to a bigram tagger.

	Accuracy
Baseline Tagger	86.76%
NLTK Bigram Tagger	88.23%
NLTK Trigram tagger	88.64%
NLTK Tagger	89.7%
HMM POS Tagger	95.68%

Table 3: Results

	RB	**V_VM**	**N_NN**	**PR_ PRP**	**PR_ PRP _M**
N_NN	6.2	3.8			
V_VM	3.2		4.9		
N_NNP			17.6		
PR_PRP					3.8
PR_ PRP _M				2.7	

Table 4: Confusion Matrix

7.5 Some Common Tagging Errors

The confusion matrix in Table 4 highlights in percentage some of the common tagging errors present in the tagger. The most common and difficult to disambiguate is when proper nouns are tagged as common nouns, and when nouns follow verbs- the tagger tags them as adverbs. Another case when verbs are tagged as nouns and vice versa are often the case of pronouns tagged as pronominal markers and vice-versa as mentioned in section 6.1.

8 Conclusion

Developing language technology tools for an under-resourced language such as Khasi has been challenging and simultaneously exhilarating to discover the nitty-gritty of the language in the way

studies such as this one exposes. The performance of the HMM tagger conditioned with the features intrinsic in the language has shown that it also provides good performance as reported in the literature relating to HMM POS taggers. This work, being a new initiative, annotating the corpus and developing the tagger, is limited by available resources; however, increasing the size of the annotated corpus for further analysis will be a good step forward.

Acknowledgement

This work was sponsored as a minor project by the University Grants Commission (UGC), India Ref.No. F.5-123/2014-15/MRP/NERO/1860.

References

E. Bars. 1973. *Khasi English Dictionary*. Shillong, Meghalaya: Don Bosco.

Sankaran Baskaran, Kalika Bali, Tanmoy Bhattacharyya, Pushpak Bhattacharyya, Monojit Choudhury, Girish Nath Jha, Rajendran S, Saravanan K, Sobha L, and KVS Subbarao. 2008. A Common Parts-of-Speech Tagset Framework for Indian Languages. *In Proceedings of LREC 2008,* (pp. 1331-1337). Marrakech, Morocco: European Language Resources Association. Retrieved from http://www.aclweb.org/anthology/I08-7013

Akshar Bharati, Dipti Misra Sharma, Lakshmi Bai, and Rajeev Sangal. 2006. *AnnCorra : Annotating Corpora Guidelines For POS And Chunk Annotation For Indian Languages*. Hyderabad: Language Technologies Research Center, IIIT Hyderabad. Retrieved from https://researchweb.iiit.ac.in/~rashid.ahmedpg08/ilmtdocs/chunk-pos-ann-guidelines-15-Dec-06.pdf

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. CA: O'Reilly Media Inc.

Thorstens Brants. 2000. TnT-A statistical part of speech tagger. *In Proceedings of the Sixth Conference on Applied Natural Language Processing* (pp. 224-231). Seattle, Washington. doi:10.3115/974147.974178

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing. A case study in part-of-speech-tagging. *Computational Linguistics*. *21(*4), 543-565. Retrieved from http://www.aclweb.org/anthology/J95-4004

Narayan Chaudhary, Pinkey Nainwani, Ritesh Kumar, and Esha Banerjee. 2010. ILCI Parts of Speech (PoS) Annotation Guidelines. Unpublished Manuscript, Indian Languages Corpora Initiative, Jawaharlal Nehru University, New Delhi, India.

Gérard Diffloth. 2018. Austro-Asiatic languages. In *Encylopaedia Britannica*. Retrieved from https://www.britannica.com/topic/Austroasiatic-languages.

Navneet Garg, Vishal Goyal, and Suman Preet. 2012. Rule based Hindi part of speech tagger. *In Proceedings of COLING*. (pp 163-174). Mumbai. Retrieved from http://www.aclweb.org/anthology/C12-3021

Tunga Güngör. Part-of-Speech Tagging. In Nitin Indurkhya & Fred J. Damerau (Eds). *Handbook of Natural Language Processing* (2nd ed., pp. 205—235). NY: Chapman & Hall/CRC, CRC Press.

Péter Halácsy, András Kornai and Csaba Oravecz. 2007. HunPos – An open source trigram tagger. *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions* (pp. 209-212). Retrieved from https://www.researchgate.net/publication/228524009_HunPos_an_open_source_trigram_tagger

Eugénie J. A. Henderson. 1976. Vestiges of morphology in modern standard Khasi. *Oceanic Linguistics Special Publications*. *13,* 477-522. Retrieved from https://www.jstor.org/stable/20019169

Daniel Jurafsky, and James H. Martin. 2009. Part-of-Speech Tagging. *Speech and Language Processing*. *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. (2nd ed, pp. 125-174). Noida: Pearson India Education

Mumtaz Bory Jyrwa. 1989. *A Descriptive study of the Noun Phrase in Khasi* (Doctoral dissertation). Retrieved from http://shodhganga.inflibnet.ac.in/handle/10603/61398

K. S. Nagaraja. 1985. *Khasi a Descriptive Analysis* Pune, India. Deccan College Post-Graduate & Research Institute.

K. S. Nagaraja. 2000. Word Formation in Khasi. *Bulletin of the Deccan College Research Institute*, 60/61, 387-417. Retrieved from http://www.jstor.org/stable/42936628

Kishorjit Nongmeikapam, and Sivaji Bandyopadhyay. 2012. A Transliteration of CRF Based Manipuri POS Tagging. *In Proceedings of 2nd International Conference on Communication, Computing & Security (ICCCS)*. *6,*(pp. 582-589). Elsevier. doi: org/10.1016/j.protcy.2012.10.070

S. Laksmana Pandian, and T.V. Geetha. 2008. Morpheme based Language Model for Part of

Speech tagging. *POLIBITS*. *38,*19-25. Retrieved from http://www.scielo.org.mx/pdf/poli/n38/n38a3.pdf

Braja Gopal Patra, Khumbar Debbarma, Dipankar Das, and Sivaji Bandyopadhyay. 2012. Part of Speech (POS) Tagger for Kokborok. *In Proceedings of COLING 2012. Posters.* (pp. 923-932). Mumbai, India. Retrieved from http://www.aclweb.org/anthology/C12-2090.

Barbara Plank, Anders Søgaard, A. and Yoav Goldberg, Y. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (pp. 412-418). Berlin, Germany. Retrieved from http://www.aclweb.org/anthology/P16-2067

Lili Rabel. 1961. *Khasi a language of Assam*. Baton Rouge: Louisiana State University Press.

Lawrence R. Rabiner. 1989. A tutorial on Hidden Markov Models and selected applications in speech recognition. *In Proceedings of the IEEE*. *77*(2), 257-285. doi: 10.1109/5.18626

H. Roberts. 2005. *A Grammar of the Khasi Language.* New Delhi, India: Mittal Publications (Original work published 1891)

M. Selvam, and A.M. Natarajan. 2009. Improvement of Rule Based Morphological Analysis and POS Tagging in Tamil Language via Projection and Induction Techniques. *International Journal of Computers*. *4*(3), 357-367. Retrieved from https://pdfs.semanticscholar.org/ffb7/494c35b766d0cac1a87298ea4f9bf00f5ad2.pdf

Manish Shrivastava, and Pushpak Bhattacharyya. 2008. Hindi POS Tagger Using Naive Stemming: Harnessing Morphological Information without Extensive Linguistic Knowledge. *International Conference on NLP (ICON 08)*. Pune, India. Retrieved from https://www.cse.iitb.ac.in/~pb/papers/icon08-hindi-pos-tagger.pdf

Smriti Singh, Kuhoo Gupta, Manish Shrivastava, and Pushpak Bhattacharyya. 2006. Morphological richness offsets resource demand – experiences in constructing a pos tagger for Hindi. *In Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*. (pp. 779-786). Sydney, Australia. Retrieved from https://www.cse.iitb.ac.in/~pb/papers/ACL-2006-Hindi-POS-Tagging.pdf

Thoudam Doren Singh, and Sivaji Bandyopadhyay. 2008. Morphology Driven Manipuri POS Tagger. *In Proceedings of IJCNLP*. (pp. 91-97). Hyderabad, India. Retrieved from http://www.aclweb.org/anthology/I08-3015

Thoudam Doren Singh, Asif Ekbal, and Sivaji Bandyopadhyay. 2008. Manipuri POS tagging using CRF and SVM: A language independent approach. *In Proceedings of 6th ICON* (pp. 240-245). Pune, India. Retrieved from http://www.academia.edu/1150196/Manipuri_POS_Tagging_using_CRF_and_SVM_A_Language_Independent_Approach

Badaplin War. 2011. *Ki Sawa bad Ki Dur Jong Ktien Khasi* (2nd ed). Shillong. Meghalaya: Ri-Ia-dor.

Richard Xiao. 2010. Corpus Creation. In Nitin Indurkhya & Fred J. Damerau (Eds). *Handbook of Natural Language Processing* (2nd ed., pp. 147—165). NY: Chapman & Hall/CRC, CRC Press.

Infant Crying Cause Recognition using Conventional and Deep Learning based Approaches

Shivam Sharma
IIIT Sri City
630 Gnan Marg, Chittoor
Andhra Pradesh, India
shivam.sharma@iiits.in

P. Viswanath
IIIT Sri City
630 Gnan Marg, Chittoor
Andhra Pradesh, India
viswanath.p@iiits.in

Vinay Kumar Mittal
Ritwik Software Technologies Pvt. Ltd.
Road No 45, Jubilee Hills
Hyderabad, Telangana, India
drvinaykrmittal@gmail.com

Abstract

Significant information about psycho-physiological state of an infant is embedded in the cry signal. Few studies are carried out for identifying and analyzing this information. But the infant cry-cause recognition task using computational classification models is not attempted yet. In our earlier works, the characterization of infant cry signals for different cry-causes were studied. In this study, we aim to recognize different cry-causes using machine-learning based classifiers. A dataset (ICSD2) was especially collected for this study. Acoustic features F_0 contour, sub-band spectral energies and MFCCs are extracted using the known signal processing methods. The infant cry-cause recognition task is explored using k-nearest neighbour (kNN), feed-forward neural network (FFNN) and convolutional neural network (CNN) classifiers. A 55-dimensional feature-set is used for kNN and FFNN classifiers. The CNN classifier is used for learning the suitable representations of MFCCs and delta MFCCs features, especially for imbalanced datasets of cry-cause categories. Performance using the FFNN classifier is better than primary classification results obtained using manual feature selection and few state-of-art results. Performance results using the MFCC features and FFNN classifier are better for cry-cause recognition. This paper shall be helpful towards infant cry-cause recognition and acoustic feature representation using deep neural networks further.

1 Introduction

The advent of the research on paralinguistic speech analysis is traced back to the middle of the 20^{th} century, notably through the statements given by Crystal, that defined it as "vocal factors involved in paralanguage" (Crystal, 1974). Although, with no formal connection to the linguistics, the potential of the meaningful information contained within the acoustics of an infant cry, has already started convincing medical practitioners, parents, care-givers, etc., about its diagnostic importance.

Major work in Paralinguistic research, for instance academic challenges are being directed towards objectives like speech emotion recognition (Schuller et al., a,b). Although these efforts streamline state specific discoveries for paralinguistic, but as an outcome they lead to limited growth for understanding relevant techniques and useful resources for the applications that do not involve linguistic information within the acoustic signal, and hence the development remains largely application specific. It is this aspect that the work done in this paper attempts to address.

Tools like short-time spectrograms and cross-correlograms have been used towards majority of the infant cry acoustic analysis, leading to spectral and inter-segmental cross-correlation analysis in (Neustein, 2010; Petroni et al., 1994; Sharma et al., 2017). Attempts involving Fundamental frequency (F_0) contour by implementing Welch's method, autocorrelation, FFT, and modZFF, have been observed to be crucial for characterising excitation source within an infant cry in (Petroni et al., 1994; Cohen and Lavner, 2012; Sharma et al., 2017; Sharma and Mittal, 2017b; Mittal, 2016b,a). ZFF along-with dominant frequency analysis is used to analyze shouted speech in (Mittal and Vuppala, 2016; Mittal and Yegnanarayana,

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 20–27

2013). Also, evaluation of base-line features like pitch, formants and MFCCs, along-with popular classification techniques like SVM and k-NN, including the neural network based classifiers like feed-forward and convolutional neural networks as well in (Galaviz and García, 2005; Reyes-Galaviz et al., 2008; Sahak et al., 2010; Zabidi et al., 2010; Cohen and Lavner, 2012; Lavner et al., 2016), have resulted in insightful observations.

Spectral information from higher order cumulants, along-with base-line features like MFCC, LPC and PLP, was observed to elucidate the non-linearity towards classifying Normal vs. Pathological cry sounds in (Chittora and Patil, 2015). Authors in (Orozco et al.) attempted to classify infant cries into Hunger or Pain, by individually evaluating linear prediction coefficients and signal intensity using a feed-forward neural network. A duration-thresholding based pre-processing step of cry sound segmentation along-with sequential forward floating feature selection approach, was taken up in (Chang et al., 2017) and compared with the results from (Abdulaziz and Ahmad, 2010) for evaluating spectro-temporal features for a dataset with 490 cry samples, towards cry-cause classification as Hunger, Lack of sleep or Pain.

A pressing concern in this field is the shortage of publicly available datasets of infant cries for categorical studies. Another direct challenge posed is about the disparateness of the categories being studied. The work has been done for a variety of causes ranging from pathologies like Asthma to disorders like Asphyxia, Ventricular septal defect (VSD), Upper respiratory tract infection (URTI), etc., in (Wahid et al., 2016; Chittora and Patil, 2015). This diversifies not only the utility of the characteristic feature-set, but also the understanding about the approaches suitable for a class specific study. All these challenges are resonated with the limitations imposed by the unavailability of infant cry dataset in public domain. It is this lack of understanding and common frameworks with respect to the resources and techniques, that the fundamental approach adopted in this work and the observations made thereof, towards infant cry acoustic analysis and cause recognition, is motivated from.

The rest of the paper is organised as follows. Signal processing methods used and features examined are stated in Section 2. Infant cry corpus collection and organization is discussed in Section 3. This is followed by the description of the experimental setup in Section 4. Acoustic analysis of the infant cries is described in Section 5. The observations related to the evaluation of crying cause classification are stated in Section 6. Section 7 provides a detailed account of the key results obtained. Finally, the paper is summarized and concluded in Section 8.

2 Signal processing methods used and Features examined

2.1 Signal processing methods used

1. *Short-time Fourier analysis:* Frequency domain processing of a signal, considered for short-time durations (Oppenheim et al., 1989).
2. *Autocorrelation:* Provides a measure of self-similarity over time (Haykin, 1989).
3. *Filter-bank spectral analysis:* Critical frequency band based spectral content analysis (Bourlard and Dupont, 1997) and (Lyons, 2012).
4. *Cepstral analysis:* Obtaining Cepstral coefficients as features representing speech sound production system characteristics in quefrency domain (Chang et al., 2017).

2.2 Features explored

1. F_0 *contour:* Functionals like Standard-deviation (dev_{F_0}) and Mean $(mean_{F_0})$ of F_0 contour are computed at the cry segment level.
2. *Sub-band spectral energy (SSE):* The ratios between the 4^{th} sub-band to 1^{st} and 2^{nd} sub-bands, i.e., $(\epsilon_{X_{4:1}})$ and $(\epsilon_{X_{4:2}})$, commonly denoted as SSE_r are computed.
3. *Mel frequency cepstral coefficients:* A total of 52 coefficients, including MFCCs $(MFCC)$, delta MFCCs $(\Delta MFCC)$, and their Standard-deviations, $(MFCC_{dev})$ and $(\Delta MFCC_{dev})$ respectively, with 13 coefficients each are computed.

3 Infant Cry Corpus (IIIT-S ICSD2)

An infant cry dataset IIIT-S ICSD2, having a total of 104 subjects (50 Male and 54 Female), age ranges for whom lie between 2 days and 6 years

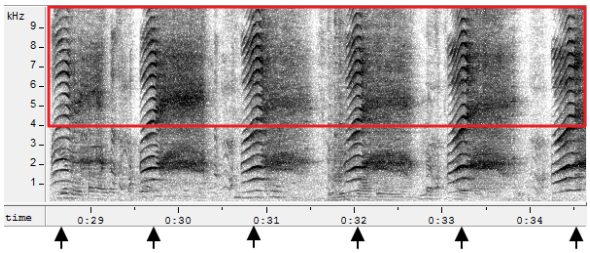


Figure 1: Spectrogram for Pain cry, depicting arched excitation contour (Marked by arrows) and consistently high spectral energy (Marked by rectangle box).

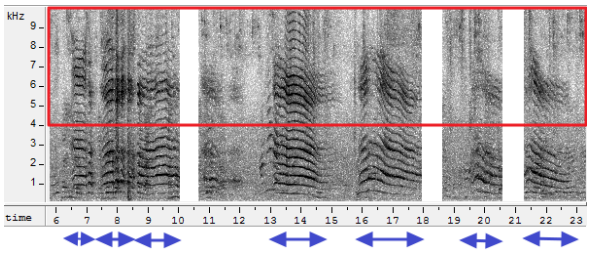


Figure 2: Spectrogram for cry due to Environmental change, depicting monotonous excitation contour (Marked by arrows) and lesser spectral energy in higher spectrum (Marked by rectangle box).

is collected for this study. From a total of 7 categories noted for the data-collection at the clinic, 4 are specifically focussed upon, for acoustic analysis and cause classification. Amongst these, *Pain* and *Stranger's anxiety* are the most prominent reasons that induced crying during the hospital visits for majority of infants. Whereas, *Discomfort* and *Environmental change* were other important cry categories for which the data is collected. On-going and historical medical conditions, parent's inputs and infant's present health status as adjudicated by the doctor, formed the basis of ground truth categorization. Due to the category-wise constraint of cry sample count in the data set, these categories are aggregated to form higher level classes, *Severe* and *Non-severe*, for the study in this work. Further corpus details can be referred from (Sharma and Mittal, 2017a).

4 Experimental setup

4.1 Acoustic Analysis

Cry signals are recorded at 48 kHz and 24 bit coding rate. The short-time analysis of the cry signal is done by considering Frame size of 30 ms and frame shift of 10 ms. Denoising of the computed fundamental period is done using median filtering of order 5. For filter-bank analysis, 6 sub-bands covering the successive spectrum ranges of 1 kHz each, starting from 100 Hz up-to 6 kHz are considered. The order of the Mel scale cepstrum is set as 13. Feature extraction and conversion into vector format is done using MATLAB R2017a and Python routines.

4.2 Cause Classification

Classifier models are implemented using statistics and machine learning, parallel processing and neural network toolboxes, in MATLAB R2017a.

The neuron count in the *Hidden layer* of the *Feed-forward neural network* has been empirically evaluated, based upon the following conventions (Heaton, 2008),

- $I_{nc} < H_{nc} < O_{nc}$,
- $H_{nc} = \frac{2}{3}(I_{nc} + O_{nc})$ and
- $H_{nc} < 2^* I_{nc}$.

where, I_{nc}, H_{nc} and O_{nc} are the neuron count for input, hidden and output layers respectively. *Convolution neural network* architecture based evaluation is done using Keras routines with Tensor flow as backend in IPython environment. Adopted from the task on *environmental sounds classification* (ESC) (Salamon and Bello, 2017), the CNN has 4 convolution layers, each followed by a relu activation layer, with max-pooling of size (2,2) and dropout after every set of 2 conv-activation layers, dropout being 15 and 20 % respectively. Next is a fully connected layer with 256 neurons with relu activation and a 50 % dropout. Finally the output layer neurons are activated using softmax function.

5 Acoustic analysis of Infant Cry

The primary experiments are focussed upon validating the acoustic characteristics observed from the cry spectrograms, as shown in Fig. 1 and 2. It is observed that the cry signals having majority of the cries with relatively *less* deviation of the F_0 contour and more stability (Fig. 2), are mostly observed for the causes that are *less severe* in nature, as can be observed from the F_0 contour plotted in the Fig. 4 (sub-plot (b)) for an Environmental change case. Cries due to *Discomfort* and *Environmental change* fall under this category. Whereas, the presence of arc-shaped excitation contours within a cry event as depicted in

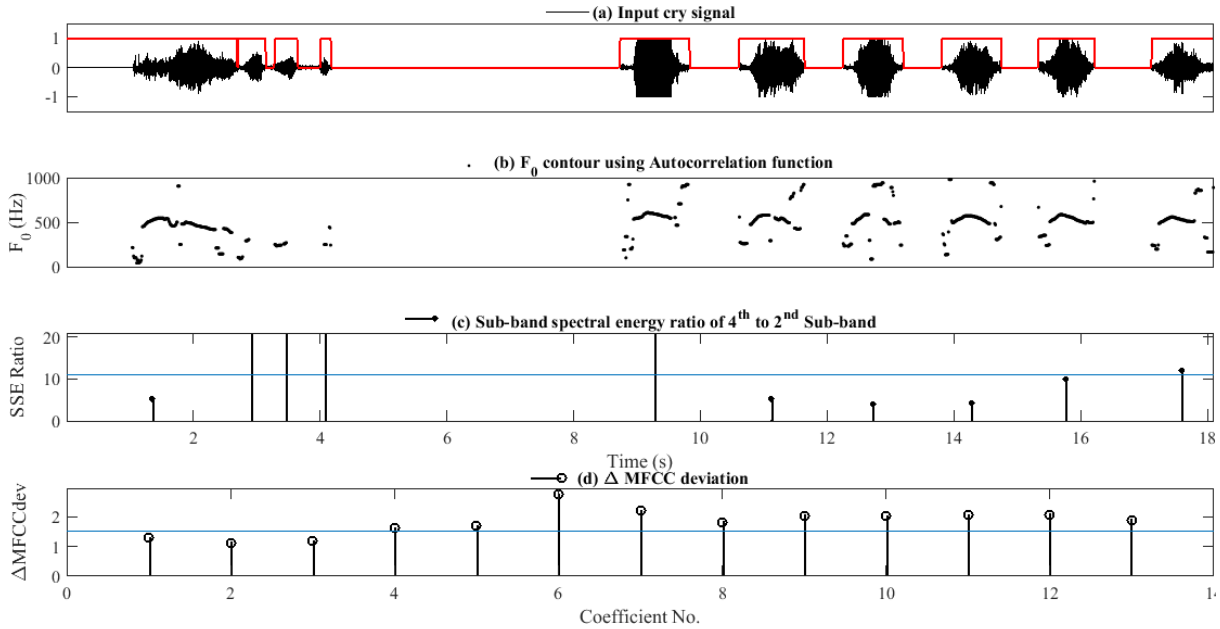


Figure 3: Comparison of (a) input signal, (b) F_0 contour (using autocorrelation), (c) Sub-band spectral energy ratio of 4^{th} to 2^{nd} sub-bands and (d) Δ $MFCC_{dev}$ for Pain cry.

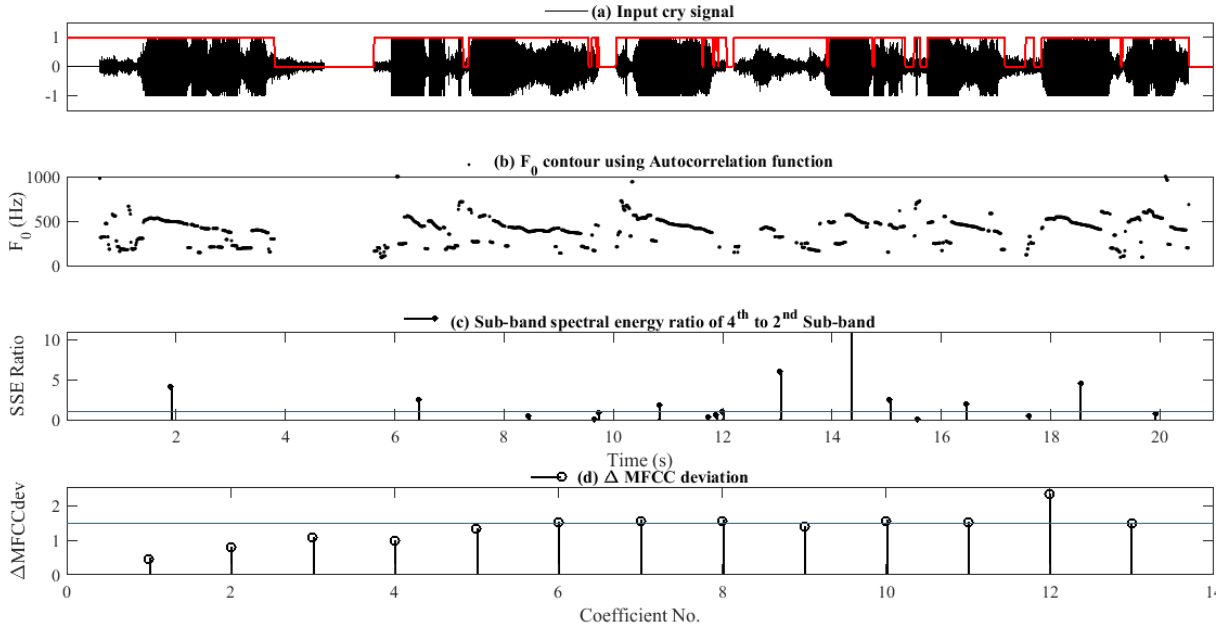


Figure 4: Comparison of (a) input signal, (b) F_0 contour (using autocorrelation), (c) Sub-band spectral energy ratio of 4^{th} to 2^{nd} sub-bands and (d) Δ $MFCC_{dev}$ for cries due to for Environmental change.

Fig. 1 indicate causes that are *severe* in nature. Cries due to *Pain* and *Stranger's anxiety* exhibit such behaviour, which can be validated from the F_0 contour of a Pain cry, as shown in Fig. 3 (subplot (b)).

For the *Pain* category, the spectral intensity appears to be more across the spectrum, as compared to that of the *Environmental change*. This effect can also be observed from the cases depicted in Fig. 1 and 2, for Pain and Environmental change respectively, wherein the spectral intensity is consistently observed to be either equivalent or more, for higher sub-bands as compared to the lower ones for *severe* categories as that for *pain*. Whereas, the same is observed to fade away for the core cry segment progressions for the *non-severe* cases like *Environmental change*. The presence of higher spectral intensity within the 4^{th}

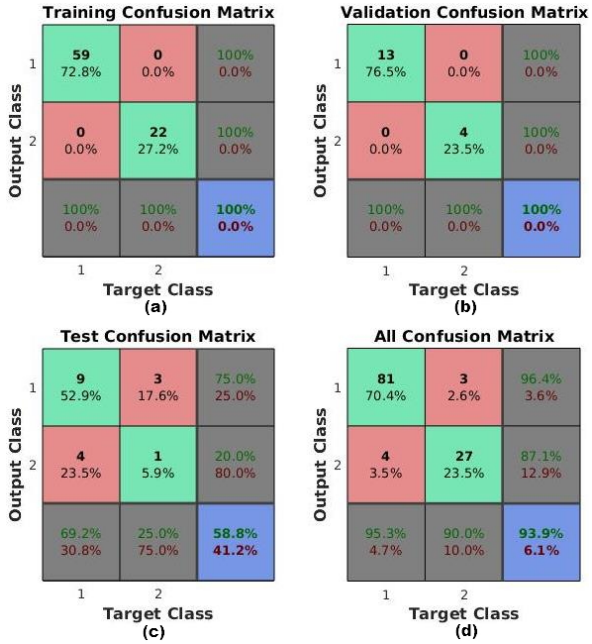


Figure 5: Confusion matrix for the 2 layered Feed-forward neural net based classification, with 90 hidden neurons.

filter-bank for *pain* category as observed from the experiments, is also demonstrated by comparing the filter-bank energy ratios of 4^{th} sub-band to 1^{st} and 2^{nd} sub-bands, for 85 Severe and 30 Non-sever cases. The average ratios for Severe category are found to be higher, at 6 and 39 respectively, whereas they are observed to be much lower at 2 and 12 for Non-severe categories. The decrease in the spectral intensity for the cries from the latter set of categories with respect to the sub-band spectral energies could possibly be attributed to the effect of *hypo-phonation* induced from the voice effects like *soft shrill*. Sub-band spectral energy ratios, plotted for the cases for Pain and Environmental change cries, as shown in Fig. 3 and 4 (sub-plots (c)) respectively, with the average ratio value for the former category being significantly *higher* than that for the latter, distinctly characterize cries as either *Severe* or *Non-severe*.

Although, infants are hardly capable of mimicking the linguistic vocalizations being actively used in their surroundings in their infancy, they do develop profound effects of paralinguistic like intonations that characterize their cultural backgrounds (Mampe et al., 2009). In an attempt to capture such time-varying spectral characteristics, MFCCs are found to divulge the cause specific characteristics effectively when subjected to distinctive classification. The qualitative difference can be easily observed by comparing the $\Delta MFCC_{dev}$ plots (Fig. 3 and 4, sub-plots (d)) for the cases from Pain and Environmental change categories, the average value of which is observed to be *higher* for *Severe* as compared to that for *Non-severe* cases. This implies greater modulations within vocalizations in the core crying regions of the bouts, captured by the dynamic functionals modelling the time-varying system characteristics.

6 Crying cause classification

6.1 Using conventional machine learning techniques

The cry-cause recognition by classification is first attempted using technique k-nearest neighbour. It is observed that $mean_{F_0}$, dev_{F_0}, SSE_r have limited performance, with relative biasing towards class Severe. Whereas, MFCC features are observed to be facilitating predictions with at least 30 % true positive rates for class Non-severe with $\Delta MFCC_{dev}$ giving the highest rate of 34 %, which is significantly greater than the performance for the rest of the features, without using any ensemble classification technique, with an overall accuracy of 80 % and true positive classification rates of 96 % and 34 % for both classes respectively, with Cosine k-NN based classification. It is important to note that this performance is obtained while evaluating the data-set with significantly skewed instance distribution, dominated by *Severe* class instances. This effect is taken into consideration by evaluating ensemble techniques like RUS boosting that takes such imbalance into consideration while evaluating the classification, which resulted in a base-line performance of 65 %. In addition to this, the problem of instance imbalance is further addressed by performing data-augmentation for audio data-set, while evaluating CNN based classification, which is discussed in further sub-sections.

6.2 Using Feed-forward neural network

Neural network based classification performs with average overall accuracy of 87.64 % with acceptable true positive prediction rates of 92.96 % and 68.68 % for the respective classes. Empirical evaluation elucidated that with the increasing no. of

Table 1: Comparison of CNN based classification performances; (a) Configurations detail format: height×width (kernel), n1-n2-n3-n4 (No. of filters in 4 layers) and (b) Datasets (Augmented).

(a) Configurations	*(b) Datasets*	*(c) Accuracy (%)*	*(c) Val. (%)*	*(d) Test (%)*
2×2, 32-32-64-64	Cry-3	57	68	63
	ESC-12	49	66	62
4×3, 12-12-16-16	Cry-3	62	60	65
Average		*56*	*65*	*63*

Table 2: Comparison of best results of several State-of-the-Art for 2/Similar class and the *Proposed* (3^{rd} observation) approach; Classifier abbreviations (c) Feed-forward neural network (FFNN), Scaled conjugate gradient (SCG) and Radial basis function network (RBFN); References in (a): [1](Orozco et al.), [2,5](Wahid et al., 2016), [3](Abdulaziz and Ahmad, 2010) and [4]*(Ours)*.

(a) Database (No. of infants)	*(b) Features*	*(c) Classifier*	*(d) Accuracy (%)*
47 Hunger, 47 Pain[1]	LPC, Intensity	FFNN (SCG)	74.70
350 Hunger, 192 Pain[2]	MFCC, LPCC, Dynamics	RBFN	86.54
88 Pain, 88 Non-Pain[3]	MFCC, LPCC	FFNN (SCG)	91.43
85 Severe, 30 Non-severe[4]	*Pitch, SSE, MFCC, Dynamics*	*FFNN (SCG)*	***93.90***
879 Deaf, 157 Normal[5]	MFCC, LPCC, Dynamics	RBFN	99.42

hidden layer neuron count and epochs, for which the neural network converges, the cross entropy error reduces. Neural network with 90 Neurons in the hidden layer outperformed all other configurations converging at 35th epoch, giving 93.9 % accuracy rate, with 95.3 % and 90 % as true positive rates for the respective classes, the confusion matrix for which can be observed from Fig. 5.

6.3 Using convolutional neural network (CNN)

Primary objective of evaluating a CNN is to examine the spatial pattern recognition capability along-with popular data-augmentation technique, while addressing the concerns of imbalanced dataset, for the task of infant cry-cause recognition. The original data sub-set is augmented 4 times using techniques like adding white noise, shifting the sound, followed by stretching using the factors 0.8 and 1.2. Also evaluated is the ESC-10 dataset (Piczak, 2015), to examine feature learning towards robust inter-class classification. Main data-sets evaluated are *Cry-3* (600), having Normal (200), Severe (200) and Non-severe (200) cry instances with 50 min. of recordings, and ESC-12 (2400) *with* Cry-3 instances along-with 9 *additional* environmental sound classes (210 min.).

Classification without any data-augmentation obviously resulted in poor performance with 34 % test accuracy. Data-augmentation helped increase the performance by approx. 30 %, which is significant. The key performances from the experimentation can be observed from the Table. 1, with configuration details specified as {*height*×*width (kernel), n1-n2-n3-n4 (No. of filters in 4 layers)*} and augmented data-sets. Second observation having *tall* filters with dimensions 4x3 and No. of filters as 12, 12, 16 and 16, with the Cry-3 (augmented) dataset, giving an overall accuracy of 62 % with 60 % validation and 65 % test accuracy outperforms other similar evaluations.

7 Results and Discussion

The results from feed-forward neural network are compared with some popular State-of-the-Art approaches (Table 2). With the available set of resources for cry analysis, a simple approach to observe and classify the infant cries as either *Severe* or *Non-severe* in this work, establishes performance up-to 93.9 %, outperforming conventional approaches and several similar/2-class classification attempts made earlier.

CNN based results do not produce optimal performance results, as against the ones already reported in (Zabidi et al., 2017), while significant

fine-tuning of network hyper-parameters still being required with the current setup. It does give insights regarding the efficacy of data augmentation for such scenarios, and using small *but taller* filters, that capture both the localized and spectral acoustic patterns, relevant in case of infant cry sounds analogous to the additional effect of temporal progression of speech utterances too.

8 Summary and Conclusion

A multi-class dataset of infant cries is used towards the infant cry-cause analysis and classification. Excitation source and production system characterising features are evaluated. It is established, that F_0 contour, sub-band spectral energy and MFCCs can distinctly characterize cries w.r.t. different causes and severity.

The significant differences in the average pitch, excitation contour patterns and spectral intensity variations across the frequency spectrum and the filter banks thereof, all for different cry-causes under consideration is established. Non-neural network based classifications yield ≈ 50 % true positive rates, which provided baseline for the current work. Whereas, MFCCs and related derivatives have shown promising performances with an average classification accuracy of 76.9 %, and also the highest accuracy of 80 % for $\Delta\ MFCC_{dev}$, suggesting the utility of the time-varying deviation in the rate of change of the *system characteristics* represented by the $MFCC$ coefficients. The required non-linearity is observed to be modelled best by the feed-forward neural networks with accuracy up-to 93.9 %. Convolutional neural networks are observed to learn discriminative feature representations that helped provide improvement upon the initial baseline obtained.

The qualitative analysis of the cry acoustics led to several observable patterns that can be refined using better cry signal processing. Such patterns are also significantly being explored within the speech recognition community that is involved with utilizing the localized spatial pattern recognition using the convolutional neural networks, but primarily towards the task of automatic speech recognition or speech emotion recognition. Acoustic signal like cry, which is devoid of any linguistic content but full of unconventional non-linguistic utterances, can also be investigated using such techniques.

Acknowledgments

The authors are thankful to Dr. Nizam (M.B.B.S-DCH), Dr. Bhavya (M.D.-Ped.) and Dr. Venkat (M.D.-Ped.), the nursing staff of Pranaam Hospital, Madinaguda, Hyderabad, and parents of the infants for supporting in collection of the cry samples needed for the study.

References

Y. Abdulaziz and S. M. S. Ahmad. 2010. Infant cry recognition system: A comparison of system performance based on mel frequency and linear prediction cepstral coefficients. In *2010 International Conference on Information Retrieval Knowledge Management (CAMP)*, pages 260–263.

Hervé Bourlard and Stéphane Dupont. 1997. Subband-based speech recognition. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 2, pages 1251–1254. IEEE.

Chang, Chang Chuan-Yu, Chuan-Wang, Kathiravan, Lin S, Chen, Chen, and Szu-Ta. 2017. Dag-svm based infant cry classification system using sequential forward floating feature selection. *Multidimensional Systems and Signal Processing*, 28(3):961–976.

A. Chittora and H. A. Patil. 2015. Classification of normal and pathological infant cries using bispectrum features. In *2015 23rd European Signal Processing Conference (EUSIPCO)*, pages 639–643.

Rami Cohen and Yizhar Lavner. 2012. Infant cry analysis and detection. In *Electrical & Electronics Engineers in Israel (IEEEI), 2012 IEEE 27th Convention of*, pages 1–5. IEEE.

D Crystal. 1974. In: Sebeok, t. (ed.), paralinguistics. In *Current Trends in Linguistics*, volume 12, page 265295, The Hague. Mouton.

Orion Fausto Reyes Galaviz and Carlos Alberto Reyes García. 2005. Infant cry classification to identify hypo acoustics and asphyxia comparing an evolutionary-neural system with a neural network system. In *MICAI 2005: Advances in Artificial Intelligence*, pages 949–958, Berlin, Heidelberg. Springer Berlin Heidelberg.

Simon Haykin. 1989. *An Introduction to Analog and Digital Communications*. John Wiley & Sons, Inc., New York, NY, USA. Pp. 652.

Jeff Heaton. 2008. *Introduction to Neural Networks for Java, 2Nd Edition*, 2nd edition. Heaton Research, Inc.

Y. Lavner, R. Cohen, D. Ruinskiy, and H. Ijzerman. 2016. Baby cry detection in domestic environment

using deep learning. In *2016 IEEE International Conference on the Science of Electrical Engineering (ICSEE)*, pages 1–5.

James Lyons. 2012. Mel frequency cepstral coefficient (mfcc) tutorial.

Birgit Mampe, Angela D. Friederici, Anne Christophe, and Kathleen Wermke. 2009. *Newborns' Cry Melody Is Shaped by Their Native Language*, volume 19.

V. K. Mittal. 2016a. Discriminating features of infant cry acoustic signal for automated detection of cause of crying. In *2016 10th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pages 1–5.

V. K. Mittal. 2016b. Discriminating the infant cry sounds due to pain vs. discomfort towards assisted clinical diagnosis. In *Proc. SLPAT 2016 Workshop on Speech and Language Processing for Assistive Technologies*, pages 37–42.

V. K. Mittal and A. K. Vuppala. 2016. Changes in shout features in automatically detected vowel regions. In *2016 International Conference on Signal Processing and Communications (SPCOM)*, pages 1–5.

V. K. Mittal and B. Yegnanarayana. 2013. Effect of glottal dynamics in the production of shouted speech. *The Journal of the Acoustical Society of America*, 133(5):3050–3061.

Amy Neustein, editor. 2010. *Advances in Speech Recognition: Mobile Environments, Call Centers and Clinics*. Springer, New York.

A. V. Oppenheim, R. W. Schafer, and J. R. Buck. 1989. *Discrete-time signal processing*, volume 2. Prentice-hall Englewood Cliffs. Pp. 71.

Jose Orozco, Carlos A. Reyes Garcia, Luis Enrique Erro, and Tonantzintla Puebla Mxico. Detecting pathologies from infant cry applying scaled conjugate gradient neural networks.

M. Petroni, M.E. Malowany, C.C. Johnston, and B.J. Stevens. 1994. A new, robust vocal fundamental frequency (f0) determination method for the analysis of infant cries. In *Computer-Based Medical Systems, 1994., Proceedings 1994 IEEE Seventh Symposium on*, pages 223–228.

Karol J Piczak. 2015. Environmental sound classification with convolutional neural networks. In *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*, pages 1–6. IEEE.

O. F. Reyes-Galaviz, S. D. Cano-Ortiz, and C. A. Reyes-Garca. 2008. Evolutionary-neural system to classify infant cry units for pathologies identification in recently born babies. In *2008 Seventh Mexican International Conference on Artificial Intelligence*, pages 330–335.

R. Sahak, W. Mansor, Y. K. Lee, A. I. Mohd Yassin, and A. Zabidi. 2010. Orthogonal least square based support vector machine for the classification of infant cry with asphyxia. In *2010 3rd International Conference on Biomedical Engineering and Informatics*, volume 3, pages 986–990.

Justin Salamon and Juan Pablo Bello. 2017. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283.

Bjorn Schuller, Stefan Steidl, and Anton Batliner. a. The interspeech 2009 emotion challenge. In *INTERSPEECH*, pages 312–315. ISCA.

Bjorn Schuller, Stefan Steidl, Anton Batliner, Alessandro Vinciarelli, Klaus R. Scherer, Fabien Ringeval, Mohamed Chetouani, Felix Weninger, Florian Eyben, Erik Marchi, Marcello Mortillaro, Hugues Salamin, Anna Polychroniou, Fabio Valente, and Samuel Kim. b. The interspeech 2013 computational paralinguistics challenge: social signals, conflict, emotion, autism. In *INTERSPEECH*, pages 148–152. ISCA.

S. Sharma and V. K. Mittal. 2017a. Infant cry analysis of cry signal segments towards identifying the cry-cause factors. In *TENCON 2017 - 2017 IEEE Region 10 Conference*, pages 3105–3110.

S. Sharma and V. K. Mittal. 2017b. A qualitative assessment of different sound types of an infant cry. In *2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON)*, pages 532–537.

Shivam Sharma, Pruthvi R. Mykala, Rajasree Nalumachu, Suryakanth V. Gangashetty, and Vinay Kumar Mittal. 2017. A study on acoustic features of infant cry signal for different causes of crying. *3rd international workshop on Affective Social Multimedia Computing (ASMMC), INTERSPEECH 2017*. Stockholm.

N. S. A. Wahid, P. Saad, and M. Hariharan. 2016. Automatic infant cry classification using radial basis function network. In *Journal of Advanced Research in Applied Sciences and Engineering Technology*, volume 4 of *1*, pages 12–28. ISSN (online): 2462-1943.

A. Zabidi, W. Mansor, L. Y. Khuan, I. M. Yassin, and R. Sahak. 2010. The effect of f-ratio in the classification of asphyxiated infant cries using multilayer perceptron neural network. In *2010 IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, pages 126–129.

A Zabidi, Ihsan Yassin, H A. Hassan, Nurlaila Ismail, M M. A. M. Hamzah, Zairi Rizman, and Husna Zainol Abidin. 2017. Detection of asphyxia in infants using deep learning convolutional neural network (cnn) trained on mel frequency cepstrum coefficient (mfcc) features extracted from cry sounds. 9.

Automatic Pause Boundary and Pause Duration Detection for Text-to-Speech Synthesis Systems in Indian Languages

Atish Shankar Ghone, Rachana Nerpagar, Pranaw Kumar, Bira Chandra Singh, Prakash B. Pimpale, Sasikumar M.
Centre for Development of Advanced Computing, Mumbai, pin 400049, India
{atish, rachana, pranaw, bira, prakash, sasi}@cdac.in

Abstract

Text-to-Speech synthesis systems are built using training over recorded speech and corresponding text. To achieve a natural sounding TTS system, it is very important to use correctly pause marked text data in the training phase. Similarly, pause marked text data along with duration of each pause is needed to build a pause prediction model. In this paper, we propose a method to mark the pauses automatically at appropriate positions in the text data corresponding to the recorded speech. This approach makes use of automatic speech recognition and text correction methods. With this approach, we save large amount of human effort without compromising much on the accuracy. We describe the experiments and results for three Indian languages: Hindi, Marathi, and Odia. The system can easily be extended to other languages.

1 Introduction

While developing text-to-speech synthesis (TTS) systems for a language, there is constant endeavor to achieve higher intelligibility and naturalness.

There are two widely used techniques to build a Text-to-Speech Synthesis system: 1) Unit selection based speech synthesis (Hunt and Black, 1996), and 2) Statistical parametric speech synthesis (Zen et al., 2009; Ghone et al., 2017). Time aligned prompt files are the primary input for development of both kinds of synthesis systems. Intelligibilityof TTS output primarily depends on the accuracy of these time aligned prompt files. These files are prepared using suitable segmentation algorithms like HMM (Prahallad et al., 2006), group delay (Prasad et al., 2004), hybrid segmentation (Shanmugam and Murthy, 2014), etc. and it requires speech (wave) and corresponding transcription text files as input. Hybrid segmentation works better for Indian Languages (Shanmugam and Murthy, 2014). Further, it is also observed that we get better alignment, if a grapheme notation (e.g., comma) corresponding to the silence region in speech file is inserted into the text file.

One of the important factors of naturalness of synthesized speech is the presence of pauses at appropriate places. Pause also affects the intended meaning of speech sound. Pause prediction model is used to generate TTS output with proper pauses. This model predicts the position of pauses in the text utterance, which has to be synthesized. There are two ways to build a pause prediction model: 1) Rule Based, and 2) Data Driven Statistical Method. The rule-based approach requires availability of linguistic tools like Part of Speech (POS) tagger, morphological analyzer, shallow parser, etc. However, there is a lack of these tools with reasonable accuracy for Indian languages. That is why the data driven approach is a better choice for most of the Indian languages (Ghosh and Rao, 2012
Sarkar and Rao, 2015). Pause marked text along with the duration information is used as training data to build the pause prediction model.

Apart from the above use cases, the pause marked text and the duration information would be primary resource for any research or study related to pause marking and pause duration analysis in NLP domain.

The text utterance of the speech data does not always reflect the actual pauses taken by the speaker, as while recording, the speaker takes many pauses even if there is no grapheme indication in the text file. In order to insert the grapheme notation to indicate a pause in text utterance, corresponding to the pause in recorded speech, it is necessary to carefully listen to the recorded speech and insert pause marks at appropriate places in the text manually. This is a time consuming and costly task, also prone to inconsistency. Apart from this, finding duration of the identified pause is also a tedious task.

Attempts have been made to insert pause marks in the text and find its duration by forced align-

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 28–34

ment of the speech file with the text prompts using the HMM tool (Sarkar and Rao, 2015); (Nguyen. 2015). EHMM (Prahallad et al., 2006.) is a tool present in festvox to implement it. However, it is not so promising for Indian Languages, because it requires a great amount of post processing and manual correction. Similar observation has been made by Nguyen (2015): "EHMM could deal with pause insertion, but it often failed to predict the pause appearance or pause duration in the speech corpus".

In this paper, we describe our approach to identify pause boundary in the text using Automatic Speech Recognition (ASR) and text correction methods. The ASR system is built using the same speech and text data. We split the speech file into multiple segments using silence region as a delimiter. We pass each segment into ASR System to recognize the corresponding text. Once the text for each segment is recognized, we concatenate each segment using pause marker (comma) as the separator to regenerate the text sentence. As the output of our ASR system is not expected to be accurate, we need to do post processing of regenerated text sentence to correct it. Regenerated text sentence is corrected using a devised method and original text.

We also describe our analysis to decide the minimum silence duration to be considered as a pause. The same is used as the threshold to split the speech file.

We have integrated all the components together and released the bundle as an open-source tool under GPL license. This tool is available at: https://github.com/TTS-cdac-mumbai/.

The rest of the paper is organised as follows: Section 2 describes the ASR system and section 3 discusses silence duration analysis. In section 4, system flow has been presented and results & analysis is given in section 5. Section 6 concludes the paper with a brief summary of the work and future direction.

2 Pause Marking with ASR System

We develop the ASR system with text data and speech files that are created for TTS. We use kaldi toolkit (Povey et al., 2011) to build ASR system. It uses Hidden Markov Models (HMMs) and Gaussian Mixture Models (GMMs) to model acoustic features. The language-building tool IRSTLM (Bertoldi and Cettolo, 2008) is used to build Language Model. We transformed the TTS data to the format required by kaldi toolkit. Unified parser (Baby et al., 2012) is used for lexicon preparation. It works for 9 Indian Languages, viz., Hindi, Marathi, Tamil, Telugu, Malayalam, Kannada, Guajarati, Odia, and Bengali. We split the speech files into multiple segments and pass each segment to the developed ASR system, which outputs the corresponding text. Further, the generated text segments are concatenated with pause mark (comma) inserted at the end of each segment. We observe that the output of ASR contains significant amount of errors primarily due to inaccuracy of ASR. Therefore, the ASR output is processed for correction.

For the correction process, we have two inputs: 1) Sentence generated using ASR output (we call this A) containing some errors, but with right pause mark information represented by a comma, and 2) Original sentence (we call this O) which was referred for speech recording, but without pause mark information. Using these, a correct sentence with right pause mark information is generated using the following method.

First, apply word level tokenization to the original sentence O. Using the tokenized output, word level n-grams are generated, where n starts from 1 and goes maximum up to the length of sentence. For example,

Sentence: *ABC PQR XYZ*

Generated n-grams: *ABC* (1-gram), *ABC PQR* (2-gram), *ABC PQR XYZ* (3-gram), *PQR* (1-gram), *PQR XYZ* (2-gram), *XYZ* (1-gram).

Then, the pause marked sentence A, generated using ASR output is split into independent chunks by considering the pause marker as a separator. Now each chunk is processed to find out the nearest n-gram from the list of n-grams using minimum character level edit distance as measured by Levenshtein (1966). Nearest n-grams found for all the chunk units are then combined using a pause marker (comma) to generate a correct sentence. This corrected sentence is expected to have correct text content and the pause information. The generated correct sentence is then compared with the original sentence by neglecting the pause markers. If the match indicates that the performed correction has generated the right sentence, then the process of correction ends. If the match fails, the process of correction is repeated using unmatched corrected sentence and the original sentence. In this second iteration of correction, the

only difference is in the process of finding a nearest n-gram from the list of n-grams for a chunk. This time instead of just the chunk under consideration, previous chunk is also pre-fixed to it as a context. This combined piece of string is now used to find the nearest n-gram for the purpose of correction. Upon finding the nearest n-gram, the pre-fixed chunk is removed from the same. This is expected to give us a correct n-gram, which we might have missed in the previous iteration. The following is pseudo code for the described method

Input

orSentence =Original text sentence without all-pause mark information
asrOutPut = Erroneous text sentence generated using ASR output with right pause mark

Algorithm

```
ngrams = createNgrams(orSentence)
    pPhrases = splitIntoChunks(asrOutPut)
    correctedSentence = blank
    FOR phrase IN pPhrases do
        correctedSentence = correctedSentence
                    + pauseMarker
                    + nearestNgramFromNgramList(phrase)
        IF correctedSentence without pauseMarkers doesn't
match orSentence
            asrOutPut =  correctedSentence
            pPhrases = splitIntoChunks(asrOutPut)
            correctedSentence = blank
            FOR phrase IN pPhrases do
                correctedSentence = correctedSentence
                        + pauseMarker
                        + (nearestNgramFromNgram-
List(previousPhrase+phrase)
                        - previousPhrase)
        RETURN correctedSentence
```

ASR system was trained with various duration of speech data and it is observed that it works properly even with just one hour of speech data.

Example:

- without comma marked sentence of Marathi language

मला वाटलं कां पाणी माइया हाता खालां श्वास घेत होते.

- Comma inserted using system

मला वाटलं**, ए** पाणी माइया हाता खालां, श्वास घेत होते.

- Final corrected sentence comma marked

मला वाटलं, **कां** पाणी माइया हाता खालां, श्वास घेत होते.

This algorithm works properly even there is repetition of word or phrase in sentences.

3 Silence Duration Analysis

Deciding the duration of silence, which can be considered as threshold (i.e., any silence shorter than it should not be considered as pause) is a non-trivial task. There are two ways to handle it: 1) Considering the threshold taken by other researchers, and 2) Trying to devise a method to decide threshold.

Vadapalli, et al. (2012) did an experiment by considering three different values, i.e., 25 ms, 50 ms, and 80 ms as threshold. Campione and Véronis (2002) tried to distinguish silent pause with occlusives and suggests to consider 200ms of silence duration as pause. It states: "Silent pause shorter than 200 ms are very difficult to discriminate from occlusives and taking them into account requires enormous manual effort." In order to verify the claim, we did the following analysis.

Hindi text content of five paragraphs detailed as in table 2 was prepared. Nine different speakers were asked to read aloud these contents in natural way and the same were recorded.

The recorded speech and corresponding text were given to three new persons. They were instructed to listen to the speech files and manually put pause mark on the text files wherever they feel appropriate. It is observed that minimum silence duration which was considered as pause by all three speakers is around 180-200 millisecond.

Distinction between pause and occlusive is also crucial. Occlusives are those silences which occur even within words as a beginning part of the stop consonants, e.g. in the word "vaakya", occlusive silence occurs in the beginning part of the phone [k]. We did analysis of our recorded data and found that occlusives of various lengths like, 60ms, 90ms, 136ms, 170ms are there. Some words having relatively long occlusive are given in the following table:

Word	Occlusive Point	Silence duration(ms)
वक्तव्य(vaktavya)	va-**k**tavya	170
मतदान(matdaan)	ma-**t**daan	138

नेपाल(nepaal)	ne-**p**aal	136

Table 1: Example words with occlusive

Following figures of wave files showing the occlusive silence illustrates our point.

Figure 1: Occlusive silence in the wave file for word "vaktavya"

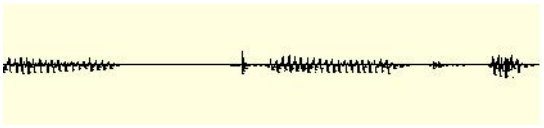


Figure 2: Occlusive silence in the wave file for the word "matadan"

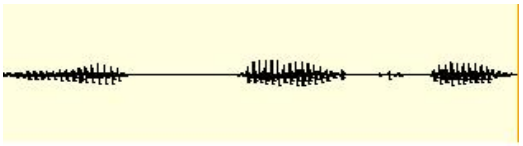


Figure 3: Occlusive silence in the wave file for the word "nepal"

In this analysis, we find two deciding factors 1) Minimum silence duration that was considered as pause by listeners and 2) Silence duration that can be distinguished clearly from occlusives. Based on these factors, we decided to consider silence duration of 200 ms as threshold to consider a pause mark.

Even for normal text, reading speech rate and duration of silence varies from speaker to speaker. Therefore, we tried to find more specific threshold for individual speaker based on the speech rate of speaker. We calculated the duration of each silence part present in our data. It was observed that there was very wide range of silence durations starting from 10 ms to 2000 ms present in the speech. We ignored the following:

- Silence durations less than 50 ms, which are generally not easily perceived.
- The longest 5% of silence durations, which are mostly exceptional cases, like 1042 ms, 1008 ms, 998 ms, 1111 ms.

Speech Rate (Syllable Rate), Arithmetic Mean (AM) and Geometric Mean (GM) of silence duration for all the 9 speakers is calculated (Table 1). From these results, no direct relation between GM and speech rate, and AM & speech rate can be established and nothing can be said about the minimum silence duration to consider as pause.

Speaker List	Arithmetic Mean (AM)	Geometric Mean (GM)	Speech Rate
Speaker 1	141.24	92.6	3.88
Speaker 2	131.87	98.2	4.59
Speaker 3	103.98	100.73	4.67
Speaker 4	135.51	105.93	4.31
Speaker 5	127.51	107.26	4.73
Speaker 6	119.06	109	5.1
Speaker 7	112.22	114.42	4.27
Speaker 8	137.17	115.12	3.74
Speaker 9	159.85	128	3.54

Table: 2 AM, GM and Speech Rate

4 System Flow

The complete system flow for pause boundary detection and pause duration, takes speech data and corresponding text file as input and generates the following outputs:

1. Text file having content with proper pause marks
2. An error report giving hint about the doubtful places, which needs to be manually verified.

The system flow is depicted in the flow chart 1.

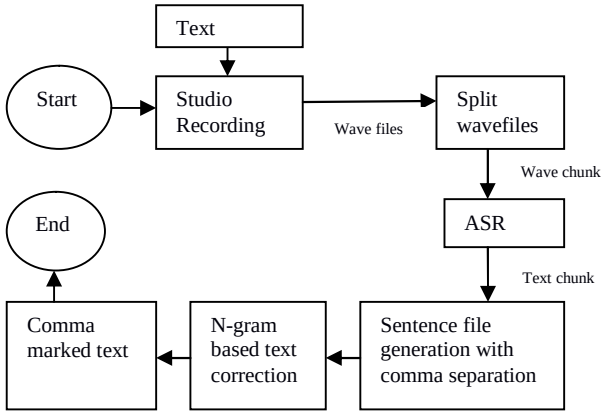


Figure 4: System flow chart

5 Results & analysis

We evaluated and analysed performance of the method in the following way.

5.1 Accuracy of text correction algorithm

As we mentioned, ASR output at sentence level is not perfect as it contains some addition, deletion, and change of words. Therefore, we apply the text correction algorithm and get significant improvement. The following table shows text improvement in two iterations. Correct sentence is the regenerated sentence, which is exactly same as original sentence (excluding pause marks).

Language	Total No of Sentences	% Correct Sentence in ASR o/p	% Correct Sentences (Iteration1)	% Correct Sentences (Iteration 2)
Hindi	2318	38	85.03	89.94
Odia	3570	49	81.45	88.01
Marathi	1889	45	89.35	93.06

Table 3: Text Correction Algorithm Accuracy

5.2 Accuracy of pause marking by the system

System has been tested with 4 sets of TTS training data:

1. Language: Marathi, Gender: Male
2. Language: Hindi, Gender: Male
3. Language: Hindi, Gender: Female
4. Language: Odia, Gender: Male

The TTS training data is available at IITM website(www.iitm.ac.in/donlab/tts/) and each set of data contains pause marked text file. System generated text file (with pause mark) is compared with manually pause marked text content. Given below is the result in Table 4:

Language	True Positive	False Negative	False Positive
Marathi male	95.51%	1.44%	3.02%
Hindi Male	91.06%	8.15%	0.38%
Hindi Female	97%	1.21%	1.3%
Odia Male	97.06	0.02%	0%

Table 4: Results

True Positive: Manually labelled as pause and identified as pause by the system
False Negative: Manually labelled as pause and identified as not pause by the system
False positive: Manually labelled as not pause and identified as pause by the system.

True Negative is not calculated, as it is not meaningful.

We observed that the text correction method is unable to correct ASR outputs with missing text for certain speech utterances. For example in the following table, example 1 shows that word 'स्त्रीची' is completely missing from ASR output and so was not corrected as expected. Whereas, for other examples we can see that 'अंधार गेल्या' is has almost similar utterance as 'सुधारलेल्या' and so the sentence was corrected as expected. The same applies to other cases 3 and 4.

In the case of sentence 5 which is also not corrected as expected, the error can be traced back to the fact that if, the n-grams generated using the original sentence contain some gram (other than the expected) more near to the erroneous word, that 'other' gram will be used as replacement. We can see that, in case 5 'ही' should have been replaced by the 'ठीक' (at edit distance 2 from ही) but as there was 'मी' (at edit distance 1 from ही) which is closer to 'ही', the 'मी' was preferred which is a wrong choice. Mechanism to deal with such errors is being investigated.

ASR output	1. प्रतिष्ठा, फक्त वीर पत्नी, अथवा वीर माता होण्यात नाही, तर वीर स्त्री होण्यात आहे. 2. अंधार गेल्या संस्कृतीची माती हरवलेल्या खजिन्याचा धातु सारखा भास शतकांचे पाहा 3. ऊस स्पश सांग तुजला सांगुन काय गेला 4. मी ही दारावरच्या बेलने दचकून उठेल उंट च्या स्वागतासाठी 5. ही माझे कपडे मी आल्यावर धुवून टाकीन
Original Sentence	1. स्त्रीची प्रतिष्ठा फक्त वीर पत्नी अथवा वीर माता होण्यात नाही तर वीर स्त्री होण्यात आहे. 2. सुधारलेल्या संस्कृतीची माती हरवलेल्या खजिन्याचा धातु सारखा भास शतकांचे पाप 3. तो स्पश सांग तुजला सांगुन काय गेला 4. तीही दारावरच्या बेलने दचकून उठेल तुमच्या स्वागतासाठी 5. ठीक माझे कपडे मी आल्यावर धुवून टाकीन
O/P of Correction Method	1. प्रतिष्ठा, फक्त वीर पत्नी, अथवा वीर माता होण्यात नाही, तर वीर स्त्री होण्यात आहे. 2. सुधारलेल्या संस्कृतीची माती, हरवलेल्या खजिन्याचा धातु सारखा भास, शतकांचे पाप. 3. तो स्पश, सांग तुजला, सांगुन काय गेला. 4. तीही, दारावरच्या बेलने, दचकून उठेल, तुमच्या स्वागतासाठी. 5. मी, माझे कपडे, मी आल्यावर धुवून टाकीन.

Table 5: Example outputs of Correction method

6 Conclusion

In this paper, we described the need of creating correctly pause marked text data with an automated process that can be used in the training phase of TTS building for achieving natural sounding high quality synthetic speech. We presented the mechanism to detect pause boundaries and duration of the pauses using ASR and customised text correction method. We also described methods to consider minimal pause duration that should work optimal. This would help those who would implement it for other languages. We described the experiments and results for Hindi, Marathi, and Odia, which looks promising. The system can readily be used for other languages and can contribute to further fine-tune and even achieve higher accuracy.

Currently, this system is marking only one type of pause. In future, we plan to work towards using multiple types of pause marks e.g., short pause, normal pause, and long pause. The basis of categorisation may be the duration of silence region. We also intend to study further to establish the relation between speech rate and silence duration threshold, which could be considered as pause mark. Along with the silence region, other factors like variation in intonation or energy may also be considered as the deciding factors for pause mark.

7 Acknowledgements

This work is carried out as a part of the research project “Development of Text to Speech Systems for Indian Languages with right pause mark Phase-II”, Ref. No. 11(7)/2011-HCC(TDIL), funded by Ministry of Electronics & Information Technology (MeitY), Govt. of India under TDIL Program. The authors would like to thank Prof Hema A Murthy from IIT Madras, Dr. Somnath Chandra and Ms. Swaran Lata from MeitY for their kind guidance and support.

References

Anandaswarup Vadapalli et al., Significance of word-terminal syllables for prediction of phrase breaks in Text-to-Speech systems for Indian languages, in *Proceedings of 8th ISCA Speech Synthesis Workshop*, Barcelona, Spain, 2013, pp. 189-194

Andrew J Hunt and Alan W. Black. 1996. Unit selection in a concatenative speech synthesis system using a large speech database. In Proceedings of IEEE International Conference Acoustics, Speech, and Signal Processing, vol. 1: pp. 373–376.

Arun Baby et al., 2012. A Unified Parser for Developing Indian Language Text to Speech Synthesizers. *International Conference on Text, Speech, and Dialogue*,Springer International Publishing

Aswin Shanmugam Subramanian and Hema A. Murthy. 2014. Hybrid Approach to Segmentation of Speech Using Group Delay Processing and HMM Based Embedded Reestimation. *INTERSPEECH*

Atish Ghone et al., 2017.TBT (Toolkit to Build TTS): A High Performance Framework to Build Multiple Language HTS Voice *at Interspeech-2017 Conference, held at Stockholm, Sweden on August 20-24, 2017*

Daniel Povey et al . 2011. The Kaldi Speech Recognition Toolkit. in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, Hilton Waikoloa Village, Big Island, Hawaii, US.

Estelle Campione and Jean Véronis. 2002. A Large-Scale Multilingual Study of Silent Pause Duration. *Speech Prosody 2002.* Aix-en-Provence, France.

Heiga Zen et al., 2009. Statistical parametric speech synthesis. *Speech Communication*, vol. 51, pp. 1039–1064

Kishore Prahallad et al., 2006. Sub-phonetic Modeling for Capturing Pronunciation Variation in Conversational Speech Synthesis. in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tolouse, France.

Krishnendu Ghosh and K. Sreenivasa Rao. 2012. Data-Driven Phrase Break Prediction for Bengali Text-to-Speech System. In *Contemporary Computing. IC3 2012. Communications in Computer and Information Science*, Vol 306. Springer, Berlin, Heidelberg.

Nicola Bertoldi and Mauro Cettolo. 2008. IRSTLM: An open source toolkit for handling large scale language models at Interspeech.

Parakrant Sarkar and K. Sreenivasa Rao. 2015. Data-driven pause prediction for synthesis of storytelling style speech based on discourse modes. *Electronics Computing and Communication Technologies (CONECCT) IEEE International Conference* pp. 1-5

Thi Thu Trang Nguyen. 2015. HMM-based Vietnamese Text-To-Speech: Prosodic Phrasing Modeling, Corpus Design System Design, and Evaluation. PhD thesis. Université Paris Sud - Paris XI

V. Kamakshi Prasad et al., 2004. Automatic segmentation of continuous speech using minimum phase group delay functions.

Vladimir*I.*Levenshtein,1966. "Binary codes capable of correcting deletions, insertions, and reversals."Soviet physics doklady. Vol. 10. No. 8.

http://www.festvox.org/download.html

https://www.iitm.ac.in/donlab/tts/database.php

Summarization of Table Citations from Text

Monalisa Dey[1], Salma Mandi[2], Dipankar Das[3],
[123]Jadavpur University, Kolkata
[1]monalisa.dey.21@gmail.com, [2]salmamandimandi@gmail.com,
[3]dipankar.dipnil2005@gmail.com,

Abstract

This paper reports our attempt to design a corpus for table content summarization, abstractive and extractive, where a table is cited in a scientific document. We have utilized 200 scientific publications in the computer science field covering 10 different domains like machine learning, automatic summarization etc. to construct the corpus. The dataset preparation for this work has been extremely daunting due to the nature of the data. The prepared dataset has been used for training, testing and evaluation. Manual annotators have been employed to validate the gold standard data in corpus. Moreover, we have also proposed two systems based on TF-IDF approach and Transition point approach to generate an extractive summarization system. The similarity score between the system generated summaries and gold standard data is calculated using standard metrics to evaluate the quality of the generated extractive summary. Finally, we have documented our observation have presented an error analysis of the system using standard metrics viz. BLEU and ROGUE.

1 Introduction

Authors use various non-textual components to represent information in a document or article. The most commonly used entities are tables to present findings or experimental results, graphical forms and figures for describing a process or presenting the output, flowcharts to depict the system flow, etc. These elements are a source of vital information and hence the importance of retrieving information from these components are increasing rapidly over the years. Moreover, often the most important experimental results and ideas in any article are presented using a table. A lot of time and effort can be saved if a researcher can understand the content of a table, without having to read the entire paper. It may also allow the researcher to examine more results that he would normally do. Consequently summarization systems play a major role in helping the reader extract critical information automatically and intelligently.

Most recently a few noteworthy contributions have been made in this area. ScienceDirect [1] is offering a table/figure preview feature in some of its articles. CiteseerX[2] is providing intelligent information extraction like figures, citations, pseudocodes etc. Consequently, summarization system becomes important to ease people in extracting the information automatically and quickly.

Although, table content summarization systems have many potential uses like summarizing a patient information from a table of symptoms and potential diseases, weather prediction form a table of daily weather reports, wikipedia infobox summarization, analysis of games like cricket,from their score tables etc., previous researches show that the challenge is finding a suitable corpus that can be used for training, testing and evaluating a table summarization system.

In order to address this issue, we have been motivated to start our study by constructing a corpus of table-content summarization in two forms, namely, extractive and abstractive summary using NLP based techniques. The techniques are, data preparation, information extraction, module building and validations. Extractive summary is obtained by extracting sentences from the article that describe the table, and abstractive summary is ob-

[1]www.sciencedirect.com/
[2]citeseerx.ist.psu.edu/

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 35–42

tained by extracting the captions associated with each table.

To construct the corpora, we have considered the following challenges and adopted feasible solutions;

A. ***How to obtain and pre-process the dataset?*** To address this challenge, we have downloaded 499 different tables from 200 computer science publications which covers various domains like Named Entity Recognition, Machine Translation, Machine Learning etc. Most of the articles are available in PDF format and hence we had to convert them to text format using for further processing purposes using PDFTextStream[3].

B. ***How to extract the table caption as abstractive summary?*** Extracting a caption is a challenge as it is written in various formats throughout different domains and writing styles. To address this issue, we have observed that a caption sentence for a table consists of FOUR parts. They are <*TABLE*>, <*INTEGER*>, <*DELIMETER*> and <*TEXT*>. Thus, to distinguish between a caption and the rest of the sentences, we propose that any sentence following the above-mentioned pattern is a caption, and the caption can form the abstractive summary of a table.

C. ***How to extract the reference text as extractive summary?*** Although captions provide details about the information in a table, it is quite possible that they might not contain enough information to assist a reader to interpret the content fully. To address this issue, we have extracted the text which is referencing the table within the document. In order to do so we have followed the same method as mentioned in the previous challenge with a few differences in the pattern. We have observed that a sentence in the vicinity of the reference sentence may provide accurate information about the context in which a table is used. Hence, we have also extracted and captured such contextually crucial sentences.

D. ***How to validate the obtained summarized output?*** The evaluation process of the system has been divided into two parts, namely, accuracy of summary identification and quality evaluation. For validating the first part, we have taken the assistance of two annotators namely, a manual annotator, A1 and our system. The Cohen's Kappa agreement analysis technique is then used to study the inter annotator agreement scores. For evaluating the quality of the summaries generated, we have employed a sentiment based similarity technique which generated a similarity score between the system generated and reference summaries which are again identified by A1. Moreover, the prepared corpus can be considered as a gold standard dataset. This is so because, to design the corpus, we are using captions and texts from the scientific publication written by the author himself.

E. ***How to present the output as a structured corpus*** To address this challenge, we have prepared an annotated corpus which contains information about various tables and their related features like abstractive summary, extractive summary, no. of rows, no. of columns etc.

The contributions of the task is to address the above-mentioned challenges, and present an annotated structured corpus with summarized output as a standard dataset for table content summarization.

The overall structure of the paper is as follows. Section 2 presents the related work carried out in this domain. Section 3 and Section 4 describe the dataset annotation in details and the model building. Section 5, Section 6 and Section 7 describe the evaluation process. Finally Section 8 describes the concluding remarks and future scope of the research.

2 Related Work

Table construction methods in free text are simple but the expressive capability is limited. The markup languages like HTML provide very flexible constructs for writers to design tables. The flexibility also shows that table extraction in HTML is harder than that in plain text. The task of table extraction from text document in (Ng et al., 1999) was recognizing table boundary,column and row. These are defined as three separate classification problem and relies on sample training texts in which the table boundaries, columns and rows have been correctly identified by human annotator.Machine learning algorithms are used to build classifiers from the training examples, one classifier per subproblem. This system is flexible and easily adoptable to text in different domain with different table characteristics. In (Wang and Hu, 2002) machine learning based approach has used for classification of table in HTML document as either genuine or non-genuine table. A set of novel features has defined which reflect

[3]7http://snowtide.com/PDFTextStream

the layout as well as content characteristics of tables. For the table detection task , the decision tree classifier is used as here features are highly non-homogeneous.They also experimented with support vector machine which shows the best performance in text categorization.However, this system misclassified a table due to the ambiguous content e.g,a table contain many hyper-links which is unusual for genuine table.This is case where layout features and the kind of shallow content features are not enough. Deeper semantic analysis would be needed in order to identify the lack of logical coherence.

An automated Table Extraction approach used in (Tengli et al., 2004) that exploits formatting cues in semi-structured HTML tables, learns lexical variants from training examples and uses vector space model to deal with non-exact matches among labels.In (Chen et al., 2000) tables are mined from large scale HTML texts. This task composed of five modules:hypertext processing,table filtering,table recognition,table interpretation and presentation of results. Table filtering module filters out impossible cases by heuristic rules.Table recognition module recognize table by the content of the cells.Table Interpretation module interpret the table attribute-value relationship either column wise or row wise.Presentation of results module results the table in a sequence of attribute-value pairs.

3 Dataset Construction

In order to prepare the corpus, we have utilized scientific articles downloaded from digital libraries. This is so because it is observed that tables play a major role in depicting results and observations within scientific papers. To the process , we have downloaded 200 papers covering 20 different type of domains in computer science ,like Automatic Summary, Machine Learning , Machine Translation etc. The average number of sentences in each document is approximately 202, excluding title, author names and section headings. Table 1 shows the statistics of the corpus. The following steps illustrate the overview of the dataset construction steps.

3.1 Caption Sentence Extraction as Abstractive Summary

A well written caption can demonstrate the content of a table coherently. Therefore, we have written python scripts (python version 2.7) for extracting the captions for all the tables. A caption can be written in various formats depending on the domain and writing style. In order to deal with this variation, we have developed a method to differentiate caption sentences from other sentences in the document. We have observed from various papers, that a caption sentence consists of 4 parts. They are <*TABLE*> which refer to the word Table, followed by <*INTEGER*>, which is an integer that refers to the table number in the paper. The integer is followed by a <*DELIMETER*> which refers to the delimiter at the end of the sentence like "."or ": ". Finally we have <*TEXT*> which is the description of the table content. If a sentence follows this pattern, we label it as a caption sentence which then forms the abstractive summary content of that table.

3.2 Relevant Sentence Extraction as Extractive summary

Although a caption describes the content of a table quite elegantly, studies have shown that captions ,on their own, are insufficient in describing an element to a reader.To handle this issue, we have observed that any table is referenced at least once in the document. Thus to obtain a more comprehensive understanding of the table under consideration, we have extracted its reference text from the corresponding scientific document. Our first step was to segment the document text into sentences. For identifying relevant sentences, we have followed the same pattern as described for caption extraction, with the difference in the fact that the delimiter part is absent in such sentences. Moreover, when a table is referenced in the document, the sentences which are within a certain proximity of the reference sentence, are very useful in describing the context in which the table is being mentioned. Keeping this in mind , we have assigned scores to each sentence depending on its proximity level and distance to the reference sentence. If the distance is within a certain threshold length (+/-1), we have considered it as an important sentence and included it in the summary.

3.3 Data Annotation

Once the gold standard abstractive and extractive summaries are generated, an annotated dataset is constructed to create a structured, well defined and easy to use corpus. Two separate approaches were employed in order to construct a gold standard dataset capable of automatic evaluation and also to evaluate the efficiency of our system generated output as extractive and abstractive summaries. Firstly, an external annotator was employed, who is well versed in the field of computer science. This annotator was asked to manually identify the abstractive caption based summary and extractive summary only from the dataset. Secondly, our system generated an output feature file with additional features besides the summaries. These features are paper ID, table ID, no. of rows in a table, no. of rows in a table, row attributes, column attributes, and Table type (numeric, text or hybrid).

3.4 Evaluating the Quality of the System

The gold standard corpus consists of 200 papers and 499 tables.The following subsection discusses the evaluation process briefly.

3.4.1 Inter Annotator Agreement

To help us with our evaluation, a manual annotator A1 is employed. Annotator A1 and our system are then each arbitrarily provided with 100 separate documents each and instructed to identify both the extractive and abstractive summaries separately. For both, a score "1" was assigned to each of the sentences included in the summary and "0", for the one's which are not. In order to have an idea about the degree of agreement , we selected 100 papers, whose output are generated by A1 and another 100 papers whose output are generated by my proposed system. These papers are then interchanged and the annotators were asked to either agree (1) or disagree (0) with the other output.

Thus at the end we had 200 papers scored by both annotators. Out of this, 20 tables were selected, containing a total of 4040 summary sentences each, for extractive and abstractive summary. This is then used for measuring the agreement score between annotator A1 and the system generated output using **Cohen's Kappa coefficient** κ which is defined as

$$\kappa = \frac{Pr_a - Pr_e}{1 - Pr_e}, \tag{1}$$

Where Pr_a is the observed proportion of full agreement between two annotators. In addition, Pr_e is the proportion expected by a chance and so indicates kind of random agreement between annotators.

The Cohen's Kappa agreement analysis provides κ = 0.83 and κ = 0.81 for extractive and abstractive summary agreement individually. Consequently, higher κ proves that the agreement is strong. The aim of this experiment was to evaluate how well the proposed method is able to identify the table content summary from the document.

3.4.2 Guidelines

The corpus contains separate folders for each of the 200 scientific papers of the computer science domain that we have processed.Table 1 gives a statistics of raw data. Within each folder, there are 6 separate files.

- CSV, which contains separate CSV files describing the content of each table in that paper.
- Annotation, which is the system generated feature file description for all tables in that paper.
- Document_PDF, which is the PDF version of the paper.
- Document_TXT, which contains the text version of the paper.
- Document_XML, which is the XML version of the paper.
- Summary, which contains the manually identified extractive and abstractive summary for each individual table of the paper.

Finally, a README file is included with the corpus, which describes each aspect of all the files.

4 System Design

We have proposed two models for generating templates that represent the extractive summary of a table. Our system produces a set of important terms, from each of the downloaded scientific papers. These terms are then used to generate an extractive summary of the tables contained in that paper. Finally, we have measured the similarity score between the generated templates and the gold standard summaries to evaluate the quality of summary in the dataset. The systems are described in the following subsection:

Paper Type	# Tables	Type: Text	Type: Numeric
Automatic Summary	50	21	29
Machine Learning	45	22	23
Machine Translation	55	19	36
NER	51	25	26
Question Answering	60	25	35
Sentiment Analysis	42	19	23
Speech Recognition	31	19	12
Text Classification	44	21	23
Text Segmentation	62	20	42
WSD	59	28	31
Total No.of Papers: 200			

Table 1: Statistics of the Corpus

4.1 TF-IDF Based System

4.1.1 Unigram Approach

These terms are selected by the following method:

–Initially a corpus is prepared from which gold standard extractive summaries for each table are extracted from that corpus.
–A set of unique words are collected from the gold standard dataset. Unique words are referred to as the words that are frequent or common in the reference summary.
–In the corpus there are 200 papers. For each table, in each paper, the TF-IDF score of all the terms are calculated excluding the stop words,non alpha-numeric characters and unnecessary punctuation. *TF* is the frequency of the term in that paper and *IDF* is the number of sentences in the paper where the term has occurred.
–Only those terms are considered that are within the set of unique words and belong to the highest scored terms for the template. These set of terms are referred to as Template for match(TS).
Each table can have multiple extractive summaries but there is only one TS for all the summaries of a particular table. So, we ranked them in order to see that with which extractive summary, the TS matched better. We have used Textual entailment method for ranking purpose.

4.1.2 Bigram Approach

A Bigram approach is also designed which takes into account a bigram instead of unigram. In this case, the TF-IDF score of a bigram is calculated in the document. But,here we have used only BLEU and ROUGE metrics for selecting terms in TS, whereas, we used Cosine Similarity, BLEU and ROUGE metrics in case of unigram. Also, all these scores have been used as a background knowledge in textual entailment which is used for ranking of the gold standard extractive summaries. In the evaluation section we have discussed which approach is the best among these two for generating extractive summaries.

4.2 Transition Point Based System

Transition Point(TP) is a frequency value that splits the vocabulary of a text into two sets of terms(low and high frequency). (Urbizagástegui, 1999) in thier paper, used the transition point(TP) to show its usefulness in text indexing. The mid-frequency terms are closely related to the conceptual content of a document.

4.2.1 Unigram Approach

A document$_i$ and its vocabulary V_i =$\{(w_j,tf_i(w_j))|w_j \in D_i\}$,where $tf_i(w_j)=tf_{ij}$,let TP_i be the transition point of D_i.A set of important terms which will represent the document D_i may be calculated as follows:

$$R_i = \{w_j|((w_j, tf_ij) \in V_i), \\ (TP_i.(1-u) \leq tf_ij \leq TP_i.(1+u))\} \quad (2)$$

where u is a value in [0,1].Some experiments presented in (Urbizagástegui, 1999) have shown that u=0.4 is a good value for this threshold.TP is obtained using the following formula:

$$TP = \frac{-1 + \sqrt{8 \times I_i + 1}}{2} \quad (3)$$

where I_1 represents the number of words with frequency equal to 1.We consider that terms whose

frequencies are closer to TP,are important terms and hence will get a high weight for summarization.All other terms will get a weight close to zero.

4.2.2 Bigram Approach

The terms selected by the above method were enriched with the words which have similar characteristics. This was done using a co-occurrence bigrams based formula in (López et al., 2007). We have divided this bigram version of the system into three subsystems, viz. **Module-I**, **Module-II**, **Module-III**.

Formally, given a document D_i made up of only these terms selected by using the TP unigram approach(R_i), the new important terms for D_i will be obtained in different way for three subsystems. We have taken bigram of each document and calculated their TF score. TF score is calculated as number of times the bigram occurs in the document.

4.2.3 Module-I:Left Approach

Module-I considers TF score of the bigram, whose value is greater than one. In each of these bigram , if the terms that belongs to R_i is in right most position, then, we have considered the left term. This term is the new term that is included for D_i. Formally, the new terms are generated according to the following expression:

$$R'_i = R_i \cup \{w' | (w_j \in R_i), (v = w' \cdot w_j), (v \in D_i), (tf_i(v) > 1)\} \quad (4)$$

4.2.4 Module-II:Right Approach

Module-II considers the right most term of the bigram when the terms in R_i is in left most position. Therefor, the new terms are obtained as follows:

$$R'_i = R_i \cup \{w' | (w_j \in R_i), (v = w_j \cdot w'), (v \in D_i), (tf_i(v) > 1)\} \quad (5)$$

4.2.5 Module-III:Left-Right Approach

Module-III is the integrated approach of Module-I and Module-II. Here, we have considered both left or right terms, whenever the terms in D_i is present in bigram. Formally,the new terms are obtained as follows:

$$R'_i = R_i \cup \{w' | (w_j \in R_i), (v = w_j \cdot w' or v = w' \cdot w_j), (v \in D_i), (tf_i(v) > 1)\} \quad (6)$$

We only used a window of size one around each term of R_i, and a minimum frequency of two for each bigram was required as condition to include new terms.

5 Evaluation

We have generated templates representing the extractive summary of a table using the above described systems. Then, we have ranked the templates in order to see which template has matched better with our gold standard summary. We have measured the similarity score between reference summary and templates using some standard metrics such as Cosine Similarity, ROUGE, BLEU.

6 Experiment and Results

In this work, we have proposed two systems, a TF-IDF based and a Transition point based, for generating extractive summaries.

In **TF-IDF based system** we have generated a Template for Matching(TS) with the highest TF-IDF scored terms. Now, the number of terms to be considered for TS solely depends on the result. Therefore, we experimented by taking variable number of such terms as shown in Table 2 and Table 3 .

Transition Point based system has two versions, the unigram and the bigram approach.The system based on unigrams, generates a set of terms whose frequency is close to the transition point. Similarity scores are then measured between the gold standard summary and system generated summary, using BLEU and ROUGE metrics. We have divided the bigram version of this system into three systems. We have measured their performance using the same similarity metrics and compared them with each other. We have also compared the unigram and bigram approach in the following section. The results are shown in Table 4.

7 Observation

In the experiment section we have mentioned that the experiments were done by changing the number of terms in Template for Matching (TS). It was observed that for smaller number of terms, the Cosine similarity and BLEU scores increased but there was a decrease in ROUGE score. If observed carefully, it can be seen that the top 10 terms give

# Terms	Cosine Similarity	BLEU	ROUGE		
			Precision	Recall	F-Measure
10	0.32	0.46	0.26	0.71	0.34
20	0.24	0.40	0.53	0.60	0.51
30	0.19	0.36	0.72	0.50	0.53

Table 2: Results obtained from TF-IDF unigram approach

# Terms	BLEU	ROUGE		
		Precision	Recall	F-Measure
10	0	0.003	0.009	0.004
20	0	0.003	0.009	0.004
30	0	0.002	0.1	0.003

Table 3: Results of TF-IDF bigram approach

Systems		BLEU	ROGUE		
			Precision	Recall	F-Measure
Unigram		0.044	0.36	0.47	0.08
Bigram	Module I	0.08	0.81	0.36	0.45
	Module II	0.11	0.79	0.39	0.46
	Module III	0.13	0.43	0.02	0.04

Table 4: Results obtained from Transition Point based system

Model	Approach	BLEU	ROGUE		
			Precision	Recall	F-Measure
TF-IDF	Unigram	0.46	0.26	0.71	0.34
	Bigram	0	0.003	0.009	0.004
TP	Unigram	0.044	0.36	0.047	0.08
	Bigram	0.13	0.43	0.02	0.46

Table 5: Comparison between TF-IDF and Transition Point

highest cosine similarity and BLEU scores. However, when the top 30 terms are considered, it was ROUGR which gave the highest score.

A comparison study has also been done between the proposed TF-IDF and Transition point based systems. The comparison is shown in Table 5 . We have considered only the best results obtained for each case. It is observed that in the TF-IDF unigram approach, BLEU score is better and in the Transition Point bigram approach, F-measure is better. But, It can be safely inferred that overall TF-IDF approach outperforms the Transition Point approach.

A set of unique words e.g. size,obtained,accuracy,lists,experiments etc. are collected from gold standard dataset for improving the quality of the generated template. We have tried to keep the words in template that belong to the set of unique words. In case of TF-IDF, we are able to include these unique words. Therefore, TF-IDF results are much better than Transition point based system.

8 Conclusion

In this work, we have presented our attempt to generate a gold standard corpus for table content summarization. While working it was found that the preparation of structured corpus was one of the greatest challenges. In the paper we have described how we have resolved all these challenges and prepared a corpus which is used for training,testing,evaluating a table summarization system. Moreover, we have also developed two models for the quality evaluation of our corpus. As a

future scope, we plan to increase the size of the corpus as well as include semantic features along with lexical we are planning to design a semantic approach based system. .

References

Hsin-Hsi Chen, Shih-Chung Tsai, and Jin-He Tsai. 2000. Mining tables from large scale html texts. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 166–172. Association for Computational Linguistics.

René Arnulfo García-Hernández, Romyna Montiel, Yulia Ledeneva, Eréndira Rendón, Alexander Gelbukh, and Rafael Cruz. 2008. Text summarization by sentence extraction using unsupervised learning. In *Mexican International Conference on Artificial Intelligence*, pages 133–143. Springer.

Marek Hlavac. 2013. stargazer: Latex code and ascii text for well-formatted regression and summary statistics tables. *URL: http://CRAN. R-project. org/package= stargazer*.

Yulia Ledeneva, Alexander Gelbukh, and René Arnulfo García-Hernández. 2008a. Terms derived from frequent sequences for extractive text summarization. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 593–604. Springer.

Yulia Ledeneva, Alexander Gelbukh, and R García Hernandez. 2008b. Keeping maximal frequent sequences facilitates extractive summarization. *Research in Computing Science*, 34:163–174.

Marina Litvak and Mark Last. 2008. Graph-based keyword extraction for single-document summarization. In *Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization*, pages 17–24. Association for Computational Linguistics.

Ming-Ling Lo, Kun-Lung Wu, and Philip S Yu. 2000. Tabsum: A flexible and dynamic table summarization approach. In *Distributed Computing Systems, 2000. Proceedings. 20th International Conference on*, pages 628–635. IEEE.

Ming-Ling Lo, Kun-Lung Wu, and Philip Shi-lung Yu. 2003. Method and apparatus for dynamic and flexible table summarization. US Patent 6,523,040.

Franco Rojas López, Héctor Jiménez-Salazar, and David Pinto. 2007. A competitive term selection method for information retrieval. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 468–475. Springer.

Hwee Tou Ng, Chung Yong Lim, and Jessica Li Teng Koo. 1999. Learning to recognize tables in free text. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 443–450. Association for Computational Linguistics.

Thanh Tam Nguyen, Quoc Viet Hung Nguyen, Matthias Weidlich, and Karl Aberer. 2015. Result selection and summarization for web table search. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pages 231–242. IEEE.

K Selçuk Candan, Huiping Cao, Yan Qi, and Maria Luisa Sapino. 2009. Alphasum: size-constrained table summarization using value lattices. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 96–107. ACM.

Ashwin Tengli, Yiming Yang, and Nian Li Ma. 2004. Learning table extraction from examples. In *Proceedings of the 20th international conference on Computational Linguistics*, page 987. Association for Computational Linguistics.

R Urbizagástegui. 1999. Las posibilidades de la ley de zipf en la indización automática. *Reporte de la Universidad de California Riverside*.

Yalin Wang and Jianying Hu. 2002. A machine learning based approach for table detection on the web. In *Proceedings of the 11th international conference on World Wide Web*, pages 242–250. ACM.

Khasi Shallow Parser

Medari Janai Tham
Computer Science & Engineering and Information Technology
Assam Don Bosco University
medaritham16@gmail.com

Abstract

A shallow parser is constructed for Khasi an Austro-Asiatic language, where noun phrases and verb phrases are identified. Formulation of what constitutes a Khasi noun phrase or verb phrase is carried out manually and marked in a corpus consisting of words already tagged with their corresponding part-of-speech tags. The parser is the first of its kind for the language, where the training corpus comprises of 24,194 chunks of noun and verb chunks out of a total of 3,997 sentences and 86,087 tokens. The approach in developing the parser is taken as a tagging problem using the Hidden Markov Model (HMM) and the results obtained have shown that a shallow parser is an appropriate first step when there is lack of information with regards to other phrases and the possible existence of lexical or syntactic mistakes in the training corpus.

1 Introduction

Shallow parsing is a process where a text is divided into non-recursive syntactical units such as noun phrases, verb phrases, etc. Non-recursive implies these phrases or chunks are non-overlapping and do not contain each other as proposed by Abney (1991). Shallow parsing has proven to be an alternative to full parsing of sentences, when only a subset of the information provided by a complete parser is sufficient for applications such as information retrieval, text summarization, etc. Further, it is also possible to enhance the corpus as and when information is available (Li and Roth, 2001), which is the current scenario with Khasi where only noun phrases and verb phrases have been annotated in the corpus. Khasi belongs to the Mon-Khmer branch of the Austro-Asiatic language family and spoken mainly in the state of Meghalaya. It is an analytic and partially agglutinative language having subject-verb-object (svo) word order, unlike the majority of Indian languages which are subject-object-verb (sov). It is not inflected and demonstrates derivational morphology in terms of affixes attached to a base word. These affixes can be easily detected and separated in any given word. The training corpus and test data are the same data set utilized in the development of a Khasi POS tagger (Tham, 2018) where the sentences have been supplemented with noun and verb phrases. The identified noun and verb chunks are non-recursive in nature and the actual constituents proposed for Khasi are given in section 3. Since the approach in constructing the parser is taken as a tagging problem, a Hidden Markov Model (HMM) part-of-speech (POS) tagger for Khasi with 95.68% accuracy (Tham, 2018) is also employed as a shallow parser, where only the data in the training corpus is altered to incorporate features relevant for parsing in the lines of Molina and Pla's (2002) shallow parser for English.

2 Related Work

Abney (1991) is credited with the introduction of the existence of chunks and he used hand crafted cascaded finite state transducers to detect chunks and clauses. However, Church's parser (1988) for detecting simple NPs can be attributed as the first statistical approach to noun phrase detection. Ramshaw and Marcus (1995) have approached chunking as a tagging problem by applying transformation based learning in detecting noun phrases and achieved recall and precision of 92% accuracy for base NP chunks while only 88% for tagging V and N chunks. Unlike Church's noun phrases (Church, 1988) which are simple, Ramshaw and Marcus

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 43–49

noun phrases include noun phrases formed with the use of conjunctions such as "and" and "or" or commas and where the possessive marker is treated as the first word of a noun phrase. Molina and Pla (2002) have also approached shallow parsing as another tagging problem by constructing what they termed as specialized HMMs where the learning and tagging procedures remain the same and adjustments have been made only to the training data and the output tags. These adjustments were carried out by varying the input and output combinations, and have reported results when there was an improvement from their baseline system comprising of the original training data and output tags. Their conclusion after comparing rule based systems, memory based systems, statistical systems, and combined systems was that combined systems gave better performances than individual systems with the exception of a Winnow system in some instances, but HMM systems also gave better performance than most systems, that too, requiring relatively less settings of information. Their overall performance reported an accuracy of $F_{\beta=1}$as 92.19.

In the Indian scenario, Singh et al. (2005) in their HMM chunker for Hindi, experimented with various tagging formats for chunk boundaries and chunk labeling and reported that for certain groups of words keeping only their POS tags improves accuracy than keeping both word and POS tag as tokens. They achieved a precision of 91.7% in chunk labelling. In the second contest conducted by International Joint Conference on Artificial Intelligence (IJCAI) on a Workshop on Shallow Parsing for South Asian Languages (Bharati and Mannem, 2007), POS and chunk annotated data comprising of 20,000 words of training data, 5000 words of development data and 5000 words of test data were provided for three Indian languages- Hindi, Bengali and Telegu. A total of eight teams participated using various approaches and PVS and Karthik (2007) is the only team that applies two learning techniques-HMM for chunk boundary detection and CRF for chunk labelling. They achieved best results in chunking for all the three languages Hindi, Bengali and Telegu with accuracy of 80.97%, 82.74% and 79.15% respectively. Tapping the morphological richness of a language, and justifying that a relatively small training corpus suffices, and avoiding the need of a large annotated corpus, a shallow parser for Marathi, Gune et al. (2010) achieved 97% accuracy for chunk identification using a 20,000 word size corpus. However, a recent noun phrase chunker for Marathi (Pawar et al., 2015) employed a CRF classifier for chunking. In this chunker, citing the lack of natural language processing (NLP) resources in India coupled with the fact that Marathi is a highly agglutinative language; the training corpus was generated automatically using Distant Supervision framework where the data is labeled according to some heuristic rules based on corpus statistics. Their reported F1 measure is 88.72%.

3 Labeling Khasi noun and verb chunks

According to Abney (1991), "a chunk consists of a single content word surrounded by a constellation of function words, matching a fixed template". As mentioned earlier, this imply chunks that are non-overlapping and do not contain each other. In this study, only noun and verb chunks have been identified. The elements of a Khasi noun chunk are similar to the noun phrases put forward by Jyrwa (1989) without the post-modifiers, while a verb chunk is taken to be the main verb itself along with any pre-modifiers such as auxiliary verbs excluding post-modifiers such as adverbs. The noun chunks excludes pronouns in the lines of Abney's (1991) definition of a chunk where pronouns are treated as orphans, and secondly, because they can also function as pronominal markers and subject enclitic (Jyrwa, 1989), for in such instances they do not syntactically function as noun chunks.

The corpus used for labeling the noun and verb chunks is a corpus annotated with part of speech tags from the BIS tagset for Khasi (Tham, 2018). The BIO labeling specified in Ramshaw and Marcus (1995) is followed for Khasi where each alphabet symbolizes the following:

B-XX: label **B** for a word starting a chunk of type XX.

I-XX: label **I** for a word inside a chunk of type XX.

O: label **O** for a word outside of any chunk.

Issues that surfaced while labeling both noun and verb chunks are highlighted below:

Basic noun phrase and the inclusion or exclusion of adjectives: According to Jyrwa (1989), the most basic noun phrase comprises of a number/gender marker also known as pronominal marker (PM), and a noun word followed by a subject enclitic (SE) as shown in example 1. Most of the abbreviations used in all the examples are in accordance with the Leipzig glossing rules[1], except when mentioned accordingly. In Khasi, pronominal markers are mandatory except in few instances where they are dropped. In example 1 the basic noun phrases present in the sentence are - *ka Banri ka*, and *ja*. However, during labeling a noun chunk, the subject enclitic has been excluded and the noun chunk is labeled up to and including the head noun, and we are left with *ka Banri* as a noun chunk. This is analogous to English noun chunks which can contain determiners and adjectives as specified in Sang et al. (2000). In Khasi, adjectives can occupy different positions in a sentence and they are included in a noun chunk only if they precede or immediately follow the noun they modify as shown in example 2 and 3. Therefore the possible pre-modifiers included in a Khasi noun chunk are demonstratives, cardinal numbers, quantifiers, pronominal markers, distributive particles, and adjectives (Nagaraja, 1985; Jyrwa 1989).

As mentioned earlier, instances where a pronominal is dropped are in vocative sentences, locative phrases and when a noun follows a verb (Jyrwa, 1989; Tham, 2018). In such cases a noun chunk comprises of the noun word without a pronominal marker.

1. ka Banri ka bam ja

   PM Banri SE eat rice

   "Banri is eating rice"

2. u diengsohphan bah

   PM jackfruit massive

   "a massive jack fruit"

3. long kaba skhem jingmut

   be REL strong mind

   "be strong minded"

Collocations of two or more nouns are part of the same noun chunk: Collocation of two or more nouns is a common phenomenon in Khasi where the actual meaning is derived from the summation of the words such as example 4. In most instances the noun(s) act as post modifier (example 5) while in some instances it acts as pre modifier(example 6). They are therefore labeled under the same noun chunk. It may be noted, that verbs tagged as nouns contribute to such collocations and hence give rise to noun chunks. Corpus analysis reflect that when a verb follows a noun it naturally becomes an element of the noun phrase comprising the noun in question as seen in examples 7 and 8, otherwise it is recommended that punctuation in the form of a comma (,) separates the noun and the following verb (example 9). However, when stylistic writing comes into play in the form of repetitions, then punctuation is not necessary as in example 10 where *sharai* a verb follows the noun *khynnah* and repeated as a stylistic element after the noun *blang*, but its attachment is to the noun *khynnah* and not the noun *blang*.

4. ka bai synniang kur

   PM cost fee clan

   "clan donation"

5. ka shuki dieng

   PM chair wood

   "wooden chair"

6. kynja kam

   type work

   "type of work"

7. shympriah thoh shun

   finger write lime

   "index finger"

8. sngi pdiang khatduh

   day accept last

   "last day of acceptance"

[1] http://www.eva.mpg.de/lingua/resources/glossing-rules.php

9. na une u lum, iohi baroh

   from this PM hill, see all

   sawdong

   around

   "from this hill, we can see all around"

10. khynnah sharai blang sharai masi

    youth serve sheep serve cows

    "shepherd"

Labeling imitative noun chunks: Imitative words are group of words where the ancestor(s) and its successor(s) are associated phonetically in their pronunciation, and they are used more for their stylistic characteristic. When the ancestor and the successor are preceded with their own pronominal marker, then both are tagged as separate noun chunks (example 11), but instances where the pronominal marker occurs only before the ancestor and not the successor, then the phrase is taken as one noun chunk (example 12). Here the POS tags attached to each word are in accordance with the BIS tagset for Khasi (Tham, 2018).

11. ka shnong ka thaw

    "village"

    ka/PR_PRP_M/B-NP shnong/N_NN/I-NP ka/PR_PRP_M/B-NP thaw/N_NN/I-NP

12. ki per soh per syntiew

    "orchard"

    ki/PR_PRP_M/B-NP per/N_NN/I-NP soh/N_NN/I-NP per/N_NN/I-NP syntiew/N_NN/I-NP

Inclusion and exclusion of the conjunction bad in a noun chunk: The conjunction *bad* is comparable to the English conjunction "and" and can also participate as an element in a noun phrase. In example 13 the conjunction is part of the noun chunk, but in example 14 it is excluded from the noun chunk because the pronominal marker precedes the second noun, indicating that acceptable pre-modifiers of noun chunks are the ones mentioned earlier without overlapping.

13. i mei bad papa

    "mother and father"

    i/PR_PRP_M/B-NP mei/N_NN/I-NP bad/CC_CCD/I-NP papa/N_NN/I-NP

14. i mei bad i papa

    "mother and father"

    i/PR_PRP_M/B-NP mei/N_NN/I-NP bad/CC_CCD/O i/PR_PRP_M/B-NP papa/N_NN/I-NP

Possessive particle la labeled as an element of a noun chunk: One of the functions of *la* is as a possessive marker (Tham, 2018), and in the training corpus it has been labeled as a member of a noun chunk because syntactically when a noun phrase is the object of a preposition *la* can occur as the first element of a noun phrase (example 15) and the same can be said of *la* when the noun phrase is the object of a verb (example 16).

15. ban wad jingiada na la ki

    to seek protection from POSS PM

    briew

    person

    "to seek protection from his own people"

16. ka kyrngah la ka khlieh

    3SGF shook POSS PM head

    "she shook her head"

Basic verb phrases: The various forms of verb phrases present in the corpus are as follows.

- A basic verb phrase can comprise only of the main verb or can also include any preceding auxiliaries. For eg. *bam* (eat) or *la bam* (have eaten).

- Instances where only an auxiliary verb exists without the main verb, the verb chunk includes only the auxiliary verb. For e.g. *long* in example 17.

- Any two consecutive verbs are taken as two separate verb chunks. For e.g. *sdang hap* (starts falling).

- The infinitive phrase comprising of the infinitive *ban* (to) up to the main verb which may include auxiliaries in between is considered as a separate verb chunk as shown in example 18.

- The inclusion and exclusion of the conjunction *bad* as part of a verb chunk is in the lines of how noun chunks include the conjunction mentioned earlier.

The rest of the tokens present in the corpus that are outside the mentioned chunks are marked with the **O** tag.

17. ki long ki briew kiba bha

    3PL are PM person REL good

    "they are good people"

    ki/PR_PRP/O long/V_VAUX/B-VP
    ki/PR_PRP_PM/B-NP
    briew/N_NN/I-NP kiba/PR_PRL/O
    bha/JJ/O

18. ka la nang ban shad

    2SGF AUX knows to dance

    "she knows how to dance"

    ka/PR_PRP/O la/V_VAUX/B-VP
    nang/V_VM/I-VP
    ban/V_VAUX_VINF/B-VP
    shad/V_VM/I-VP

4 HMM shallow parser for Khasi

Following the work of Molina and Pla (2002), where shallow parsing is considered as a tagging problem, a standard HMM algorithm has been employed in developing an HMM Shallow Parser for Khasi. Molina and Pla (2002) have put forward a specialized HMM where alterations have been made in the training corpus while the training and tagging procedure remains intact. They have attained results at par with existing approaches especially when lexical information is added and achieved best $F_{\beta=1}$ as 92.23. Similarly the Khasi HMM POS tagger (Tham, 2018) has been used as a parser where changes have been made only in the training corpus.

Statistically, given a set of input symbols I and a set of output symbols C, tagging a sentence S=s_1,s_2... s_n of n symbols where $s_j \in I \; \forall \; s_j$with output tags c_1,c_2... c_n where $c_j \in C \; \forall \; c_j$ is given by

$$argmax_C \prod_{i=1}^{n} P(s_i|c_i)P(c_i|c_{i-1} \dots c_{i-k}) \qquad (1)$$

Taking into account Markov's assumptions a second order Markov model reduces equation 1 to equation 2.

$$argmax_C (\prod_{i=1}^{n} P(s_i|c_i)P(c_i|c_{i-1}, c_{i-2})) \qquad (2)$$

The probabilities are then estimated from the training corpus using maximum likelihood estimation, and linear interpolation has been carried out to counter any data sparsity problem encountered as shown in equation 3.

$$P(c_i|c_{i-2}, c_{i-1}) = \lambda_3 \hat{P}(c_i|c_{i-2}, c_{i-1}) + \lambda_2 \hat{P}(c_i|c_{i-1}) + \lambda_1 \hat{P}(c_i) \qquad (3)$$

Here $\hat{P}(c_i|c_{i-2}, c_{i-1})$, $\hat{P}(c_i|c_{i-1})$, $\hat{P}(c_i)$ are the trigram, bigram, unigram probabilities respectively, and $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

Further, Brants (2000) deleted interpolation is used for evaluating the λs and the Viterbi algorithm (Rabiner, 1989) is utilized to ensure an optimal path is taken when selecting the sequence with the highest probability.

In this analysis, since no more than noun and verb chunks have been identified and the BIO tagging scheme is utilized, the total number of chunk tags (output symbols) is 2n+1 i.e. 5 where n is the number of chunks. On the other hand, the input symbols involved words and their corresponding POS tags leading to a huge set of symbols. As suggested by Molina and Pla (2002) a specialization function f_s can be applied on the manually tagged training data T to produce a new training data $\check{T}$. Here the specialization function (equation 4) transforms every training pair <s_i,c_i> to <$\check{s}_i$, $\check{c}_i$> and therefore it changes the set of input symbols to $\check{I}$ and the output symbols to $\check{C}$.

$$f_s : \mathrm{T} \subset (I \times C)^* \rightarrow \check{T} \subset \left(\check{I} \times \check{C}\right)^* \quad (4)$$

4.1 Testing and Evaluation

The transformations carried out for Khasi is accomplished by changing the input and output used for training. Initially for the baseline tagger only the POS tags are maintained as input symbols and the chunk tags as output symbols i.e. the training pair is $<p_i,c_i>$ where p_i is a POS tag and c_i the chunk tag. A sample input and sample output of the baseline HMM tagger is shown in example 19. The results of the tagger are then taken as the baseline results for the system (Table 1). In the next step, the POS tags are once more taken as input symbols, but the output symbols comprises of both POS tags and chunk tags (concatenated together with the period (.)). The new training pair is therefore $<p_i,p_i.c_i>$. Example 20 shows a sample output of the HMM shallow parser on the same input shown in example 19. The test data consist of 402 sentences which includes 2,210 noun and verb chunks and the tagging results are shown in Table 1, indicating that adding just POS information to the chunk category has dramatically improved the accuracy to $F_{\beta=1}$as 95.51 as compared to the baseline of $F_{\beta=1}$as 86.94.

19. V_VM RB IN N_NN N_NN RD_PUNC (input)

    V_VM/B-VP RB/O IN/O N_NN/B-NP N_NN/I-NP RD_PUNC/O (output)

20. V_VM/V_VM.B-VP RB/RB.O IN/IN.O N_NN/N_NN.B-NP N_NN/N_NN.I-NP RD_PUNC/RD_PUNC.O (output)

The individual results for noun and verb chunks are given in Table 2 and analysing the results reveals that in most cases where the chunks were not detected accurately are mainly due to the following:

- When the noun chunk is the object of the preposition and the chunk contains an adverb as the first element then it fails to identify the adverb as the starting element of the noun chunk.
- Non detection of conjunctions which are part of a noun chunk.
- As mentioned in section 3 consecutive nouns are always considered as part of the same noun chunk, but in some instances this is not true. These phrases are semantically determined, which is difficult to detect at this stage of the parser. For instance in example 21, *shipara* and *kynthei* are not part of the same noun chunk, but the tagger has placed both of them within the same noun chunk.

21. ar ngut ki khynnah shipara kynthei

    two CLF PM youth sibling girl

    bad shynrang

    and boy

    "two siblings, a girl and a boy"

- Auxiliary verbs following another auxiliary verb tend to be tagged as part of a new verb chunk when they are actually elements of the previous verb chunk.

	Precision	**Recall**	$F_{\beta=1}$
Baseline	86.38%	87.51%	86.94
Khasi Shallow Parser	94.39%	96.65%	95.51

Table 1: Results

Chunk	**Precision**	**Recall**	$F_{\beta=1}$
NP	93.4%	97.23%	95.28
VP	95.34%	96.1%	95.72

Table 2: Noun and Verb chunk results

5 Conclusion

This work has initiated a corpus of noun and verb chunks, and an HMM shallow parser for Khasi which requires Khasi text tagged with their part of speech. The details of what constitutes a noun chunk or a verb chunk were highlighted keeping in mind that identifying a Khasi noun chunk or a verb chunk from a given text is a new initiative for the language. The results of the parser are encouraging and are in

parity with what is reported for English and other Indian languages and enhancing corpus size will facilitate further testing. In future, when analysis of other Khasi phrases is available, what will remain is incorporating the acquired information only in the corpus without the need of modifying the tagging algorithm.

References

Steven Abney. Parsing by Chunks. 1991. In R. Berwick, S. Abney and C. Tenny (Eds), *Principle–based Parsing*. Kluwer Academic Publishers. MA (pp 257-278).

Thorstens Brants. 2000. TnT-A statistical part of speech tagger. *In Proceedings of the Sixth Conference on Applied Natural Language Processing* (pp. 224-231). Seattle, Washington. doi:10.3115/974147.974178.

Akshar Bharati and Prashanth R. Mannem. 2007. Introduction to Shallow Parsing Contest on South Asian Languages. *In Proceedings of the IJCAI and the Workshop On Shallow Parsing for South Asian Languages (SPSAL)* (pp. 1-8). Hyderabad. Retrieved from https://pdfs.semanticscholar.org/448a/80b1d27ad563d494fd698a77dfe09bd67bdf.pdf

Kenneth Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. *In proceedings of Second Conference on Applied Natural Language Processing* (pp. 136-143). Austin, Texas. doi: 10.3115/974235.974260

Harshada Gune, Mugdha Bapat, Mitesh M. Khapra, and Pushpak Bhattacharyya. 2010. Verbs are where all the action lies: experiences of shallow parsing of a morphologically rich language. *In Proceedings of the 23rd International Conference on Computational Linguistics: Posters* (pp. 347-355). Beijing. Retrieved from http://www.aclweb.org/anthology/C10-2040

Mumtaz Bory Jyrwa. 1989. *A Descriptive study of the Noun Phrase in Khasi* (Doctoral dissertation). Retrieved from http://shodhganga.inflibnet.ac.in/handle/10603/61398

Xin Li and Dan Roth. 2001. Exploring evidence for Shallow Parsing. 2001. *Proceedings of the Conference on Computational Natural Language Learning*. Toulouse. doi:10.3115/1117822.1117826

K.S. Nagaraja. 1985. *Khasi a Descriptive Analysis* Pune, India. Deccan College Post-Graduate & Research Institute.

Antonio Molina and Ferran Pla. 2002. Shallow Parsing using Specialized HMMs. *Journal of Machine Learning Research. Introduction to Special Issue on Machine Learning Approaches to Shallow Parsing*. 2, 595-613. Retrieved from http://jmlr.org/

Sachin Pawar, Nitin Ramrakhiyani, Girish K. Palshikar, Pushpak Bhattacharyya, and Swapnil Hingmire. 2015. Noun Phrase Chunking for Marathi using Distant Supervision. *In Proceedings of the 12th International Conference on Natural Language Processing* (pp. 29-38). Trivandrum. Retrieved from http://www.aclweb.org/anthology/W15-5905

Avinesh PVS and G Karthik. 2007. Part-of-speech tagging and chunking using conditional random fields and transformation based learning. *In Proceedings of the IJCAI and the Workshop On Shallow Parsing for South Asian Languages (SPSAL)* (pp. 21-24). Hyderabad. Retrieved from https://pdfs.semanticscholar.org/448a/80b1d27ad563d494fd698a77dfe09bd67bdf.pdf

Lawrence R. Rabiner. 1989. A tutorial on Hidden Markov Models and selected applications in speech recognition. *In Proceedings of the IEEE*. *77*(2), 257-285. doi: 10.1109/5.18626.

Tjong Kim Sang, Erik F. and Sabine Buchholz. Introduction to the CoNLL-2000 Shared Task: Chunking. 2000. *In Proceedings of CoNLL-2000 and LLL-2000* (pp. 127-132). Lisbon. doi:10.3115/1117601.1117631

Akshay Singh, S M Bendre and Rajeev Sangal. 2005. HMM chunker for Hindi. *Proceedings of IJCNLP*: *The Second International Joint Conference on Natural Language Processing*, Jeju Island. Retrieved from http://aclweb.org/anthology/I05-2022

Medari J. Tham. 2018. Challenges and Issues in Developing an Annotated Corpus and HMM POS Tagger for Khasi. *The 15th International Conference on Natural Language Processing.* (forthcoming). Patiala, Punjab.

Resolving Actor Coreferences in Hindi Narrative Text

Nitin Ramrakhiyani Swapnil Hingmire Sachin Pawar

Sangameshwar Patil Girish K. Palshikar

{nitin.ramrakhiyani,swapnil.hingmire,sachin7.p}@tcs.com

{sangameshwar.patil,gk.palshikar}@tcs.com

TCS Research, Tata Consultancy Services, India

Pushpak Bhattacharyya

pb@cse.iitb.ac.in

IIT Patna, India

Vasudeva Varma

vv@iiit.ac.in

IIIT Hyderabad, India

Abstract

An important aspect of understanding narrative text is identification of actors, its mentions and coreferences among them. Coreference Resolution in Hindi is a relatively under-explored area. In this paper, we focus on the task of resolving coreferences of actor mentions in Hindi narrative text. We propose a linguistically grounded approach for the task using Markov Logic Networks (MLN). Our approach outperforms two strong baselines on a publicly available dataset and 4 other manually created datasets.

1 Introduction

Narrative text describes related sequences of events involving a set of actors and interactions among them. The first step towards understanding of narrative text is to identify the actors involved and to resolve their coreferences. We define an actor to be an entity of type PERSON, LOCATION, or ORGANIZATION. These actors are referred in text through their mentions which can be of three types: named entities, *generic noun phrases (NPs)*[1] and pronouns (Walker et al., 2006).

In this paper, we aim to resolve coreferences of actor mentions in Hindi narrative text. We assume availability of gold-standard actor mentions and their types; and focus only on resolving coreferences among actor mentions. Unlike much of the earlier work, we do not restrict the coreferences to only pronouns (Anaphora Resolution) and their nominal antecedents. In addition to pronouns, we also consider generic NPs for coreference resolution. For instance, referring to the sample narrative in Table 1, we want to identify that various mentions like the named entities (सरदार पटेल, पटेल), the pronouns (वे , उन्होंने) as well as the generic NP (चौथी संतान) all refer to the same actor (सरदार पटेल).

[1] A generic NP is a noun phrase which has a common noun as its head-word.

[सरदार पटेल]$_{A_1}$ का जन्म [गुजरात]$_{A_2}$ में 1875 में हुआ था। [वे]$_{A_1}$ [झवेरभाई पटेल]$_{A_3}$ एवं [लाडबा देवी]$_{A_4}$ की [चौथी संतान]$_{A_1}$ थे। [लन्दन]$_{A_5}$ जाकर [पटेल]$_{A_1}$ ने बैरिस्टरी की पढाई की और वापस आकर [अहमदाबाद]$_{A_6}$ में वकालत करने लगे। [महात्मा गांधी]$_{A_7}$ के विचारों से प्रेरित होकर उन्होंने [भारत]$_{A_8}$ के स्वतन्त्रता आन्दोलन में भाग लिया। स्वतन्त्रता आन्दोलन में [सरदार पटेल]$_{A_1}$ का सबसे पहला और बड़ा योगदान [खेडा]$_{A_9}$ संघर्ष में हुआ। [किसानों]$_{A_{10}}$ ने [अंग्रेज सरकार]$_{A_{11}}$ से भारी कर में छूट की मांग की। जब यह स्वीकार नहीं किया गया तो [सरदार पटेल]$_{A_1}$, [गांधीजी]$_{A_7}$ एवं [अन्य लोगों]$_{A_{12}}$ ने [किसानों]$_{A_{10}}$ से मुलाकात की और [उन्हे]$_{A_{10}}$ कर न देने के लिये प्रेरित किया।

Table 1: Sample Hindi narrative. Actor mentions are marked with [. . .] and mentions of the i^{th} actor are denoted by the subscript A_i.

Coreference resolution is known to be a challenging NLP problem. Even for languages such as English, which have good quality linguistic resources and datasets, coreference resolution has proved to be a hard problem (Ng, 2017). For languages which are relatively resource-poor such as Hindi, the problem gets exacerbated.

Some of the approaches in coreference resolution in Hindi are adapted from coreference resolution approaches for English. For example, Dutta et al. (2008) adapt the well-known Hobb's algorithm for Hindi. Certain approaches involve application of linguistic knowledge for co-reference resolution. Agarwal et al. (2007) propose an approach based on matching constraints for the grammatical attributes of different words while Prasad and Strube (2000) and Uppalapu and Sharma (2009) apply centering theory (Grosz et al., 1995) for

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 50–58

coreference resolution. Dakwale (2014) proposes a hybrid approach based on dependency structure and linguistics constraints to resolve pronominal references. Mujadia (2017) proposes a hybrid approach based on Paninian dependency grammar, linguistic rules and resources like DBPedia and word-embeddings to resolve nominal coreferences.

Note that major focus of the work in coreference resolution for Hindi has been on anaphora resolution. Anaphora resolution is a subset of more general coreference resolution problem. Anaphora resolution focuses on connecting pronouns to their antecedents which refer to the same entity. It does not focus on connecting a generic NP to its antecedent(s). Further, prior work for coreference resolution in Hindi uses supervised learning algorithms which need labeled training data to induce the classifier(s).

To resolve coreferences among actor mentions in Hindi narratives, we develop unsupervised algorithms based on Markov Logic Networks (MLN) (Domingos and Lowd, 2009). MLNs combine first order logic rules with probabilistic graphical models in a single representation. We encode linguistic knowledge relevant to coreferences in an MLN and use the inference in the MLN for coreference resolution. Thus, the approach is unsupervised, avoiding the need for labeled training data.

Major contributions of this work are: i) To the best of our knowledge, this is the first attempt at actor coreference resolution for Hindi narrative text, ii) An unsupervised approach based on Markov Logic Networks for coreference resolution in Hindi, and iii) A set of robust linguistic rules encoded in MLN, despite the absence of good NLP pre-processing tools (e.g., no constituency parser or semantic role labeller for Hindi). The paper is organized as follows: Section 2 describes the related work, Section 3 covers the details of our MLN-based coreference resolution approach, Section 4 describes the experimental analysis and Section 5 concludes the work with some pointers on future work.

2 Related Work

Coreference resolution is an extensively studied problem in computational linguistics. Several authors have proposed methods for coreference resolution. These methods can be broadly classified into three types of approaches: i) rule based methods, ii) machine learning based methods and iii) hybrid methods.

Rule based methods like the Hobb's algorithm (Hobbs, 1986) represent linguistic knowledge about coreference in the form of rules which are then used for coreference resolution. These linguistically motivated rules try to model various factors of coreference resolution such as gender agreement, number agreement, semantic relations like IS-A, semantic similarity, proximity or theories like centering theory based choice of referring expression (Grosz et al., 1995). A key limitation of such rule based approaches is that they require extensive human efforts to represent and process linguistic knowledge.

Machine learning based methods on the other hand are "knowledge-poor" methods (See (Ng, 2017) for an overview). These methods use a labelled corpus to train models for coreference resolution. Recently, several authors have proposed neural methods of coreference resolution, e.g. (Clark and Manning, 2016; Lee et al., 2017). Though, neural methods have shown promising coreference resolution results as compared to other learning methods, they need a large amount of labelled data and computational resources. Hence, they can not be applied to low-resource Indian languages for which a large coreference annotated data is expensive to obtain.

In the context of coreference resolution in Hindi texts several authors adapted methods for coreference resolution in English. Dutta et al. (2008) adapted the Hobb's algorithm, while Prasad and Srube (2000) and Uppalapu and Sharma (2009) adapted centering theory based coreference resolution for Hindi. Dakwale (2014) proposes a hybrid approach which first applies a set of rules on syntactic information of sentences and then incorporates grammatical and semantic information into supervised learning methods to resolve more ambiguous instances. It is important to note that most of the work for coreference resolution in Hindi are focused on resolution of pronominal references and not on generic NPs discussed earlier. Recently, Mujadia (2017) proposes a sieve-based hybrid approach for coreference resolution of pronouns as well as nominal references. The approach uses a set of sieves using Paninian Dependency Grammar, POS labels, morphology and animacy features, linguistic resources

like Hindi WordNet, DBpedia derived named dictionary, Word2Vec and GloVe based word embeddings. However, this approach is supervised and hence, needs a labelled dataset.

A recent work by Patil et al. (2018) is the closest to our proposed approach. They use an MLN-based approach for resolving actor mention coreferences in English narrative text. They build upon the output from Stanford CoreNLP coreference resolution, whereas we attempt to address the problem from scratch.

3 Coreference Resolution using MLN

We propose a linguistically motivated approach for resolving coreferences. As it is difficult and effort-intensive to develop annotated datasets for Hindi Coreference resolution, we develop an unsupervised approach using Markov Logic Networks (MLN). MLNs combine logic with probabilistic graphical models. MLN allows representation of linguistic knowledge characterizing coreferences, in the form of weighted first order logic rules. Weight associated with each rule represents its strength. An MLN is constructed for a narrative which encodes multiple pieces of information regarding actor mentions in the narrative. It also includes the first order logic rules encoding the linguistic knowledge. Inference in such an MLN leads to the most likely coreference links among the actor mentions, while ensuring maximum weighted satisfiability of the linguistic rules.

3.1 Predicates

We design several *predicates* which are needed to represent the linguistic knowledge in the form of first order logic rules. There are two types of predicates:

- **Evidence predicates**: These predicates encode observed information regarding actor mentions and their relationships in a given narrative. Truth values are known for all groundings of such predicates.
- **Query predicates**: Truth values for all or some of the groundings of these predicates are unknown. Inference in MLN is needed to know the most likely truth values for these predicates.

Table 2 describes in detail various predicates used in the MLN rules.

3.2 Linguistic Rules

We express linguistic knowledge characterizing coreferences in the form of first order logic rules in the MLN. In addition to these rules, *evidences* are provided to the MLN in the form of observed true groundings of all evidence predicates in a given narrative.

Ensuring Equivalence of Coreferences: The query predicate $Coref$ represents the coreference relations among actor mentions, which is required to be an equivalence relation. Hence, we include following 3 rules:

Reflexivity: $Coref(x, x)$.

Symmetry: $Coref(x, y) \Rightarrow Coref(y, x)$.

Transitivity: $Coref(x, y) \wedge Coref(y, x) \Rightarrow Coref(x, z)$.

Actor Type Consistency: A necessary condition for any two actor mentions to be coreferences of each other is that their Actor/Entity types should be same. E.g., संतान *(child)* and समिती *(committee)* can never be coreferences of each other because actor type of the former is PERSON whereas actor type of the later is ORGANIZATION. The rule is expressed in first order logic as:

$$NER(x, t) \wedge NER(y, w) \wedge (t \neq w) \Rightarrow \neg Coref(x, y)$$

Identical Actor Mentions: If $Identical(x, y)$ is true for any pair of actor mentions, then x and y are likely to be coreferences. This is a high confidence rule and hence is associated with infinite weight.

$$Identical(x, y) \Rightarrow Coref(x, y).$$

Actual pairs of such actor mentions are provided as evidences to MLN. E.g., In the sentence सरदार पटेल स्वतंत्रता सेनानी थे। *(Sardar Patel was a freedom fighter.)*[2], the actor mention स्वतंत्रता सेनानी *(freedom fighter)* is **predicative nominal** of the subject सरदार पटेल. Such actor mentions generally refer to the same real life actor and these are often connected through a copula verb. The evidence provided here is:

$$Identical(\text{सरदार पटेल}, \text{स्वतंत्रता सेनानी})$$

There are some other copula-like verbs (such as बने *(became)*) which connect coreferent actor mentions. E.g., सरदार पटेल उप-प्रधानमंत्री बने। *(Sardar Patel became the Deputy Prime Minister.)* Here, the evidence would be:

$$Identical(\text{सरदार पटेल}, \text{उप-प्रधानमंत्री})$$

[2]The English translations are provided only for reading help, the rules have to be interpreted for Hindi sentences only.

Evidence predicates:	
$NER(x,t)$	True iff actor mention x is of type t
$PronounLike(x)$	True iff actor mention x is a pronoun or a definite mention
$Identical(x,y)$	True iff actor mentions x and y appear in a linguistic relationship which indicates that x and y are likely to be coreferences of each other
$LexSim(x,y)$	True iff actor mentions x and y lexicall similar, i.e. they have little edit distance or one is suffix/prefix of another
$NonIdentical(x,y)$	True iff actor mentions x and y appear in a linguistic relationship which indicates that x and y are not likely to be coreferences of each other
$IsAntecedent(x,y)$	True iff actor mention y is an antecedent for x which is a pronoun or a definite mention
Query predicates:	
$Coref(x,y)$	True iff actor mentions x and y are coreferences of each other

Table 2: Predicates used in MLN rules

Lexically Similar Mentions: If x and y are lexically similar then they are likely to be coreferences. This rule is associated with a finite weight of 10 (empirically decided using a development dataset).

$$10.0\ LexSim(x,y) \Rightarrow Coref(x,y).$$

Actual pairs of such actor mentions are provided as evidences to the MLN. Following is an example of such an evidence:

$LexSim$(सरदार पटेल, सरदार वल्लभभाई पटेल)

Non-identical Actor Mentions: If $NonIdentical(x,y)$ is true for a pair of actor mentions (x,y), then x and y are **unlikely** to be coreferences. This is a high confidence rule and hence is associated with infinite weight.

$$NonIdentical(x,y) \Rightarrow \neg Coref(x,y).$$

Actual pairs of such actor mentions are provided as evidences to MLN. There are several cases for identifying non-identical pairs of actor mentions.

1. **Conjunctions**: When two actor mentions are conjunctions of each other, then it is highly unlikely that they are coreferences of each other. E.g., सरदार पटेल और अन्य नेताओं ने गांधीजी से मुलाकात की। *(Sardar Patel and other leaders met Gandhiji.)* Here, evidence provided to the MLN is:

   $NonIdentical$(सरदार पटेल, अन्य नेताओं)

2. **Consecutive actor mentions**: If any two actor mentions appear consecutively in a sentence, then these mentions are less likely to be coreferences of each other unless both are nominal mentions. E.g., उन्होंने किसानों की समस्याओं को समझा। *(He understood the difficulties of the farmers.)*

   $NonIdentical$(उन्होंने, किसानों)

   E.g., वे उनके अग्रज थे। *(They were his elder brothers.)*

   $NonIdentical$(वे, उनके)

3. **Noun modifiers**: If an actor mention is modifying (connected through a "nmod" dependency relation) another actor mention, then it is unlikely that these mentions are coreferences of each other. E.g., वे झवेरभाई पटेल की चौथी संतान थे। *(He was Jhaverbhai Patel's fourth child.)*

   $NonIdentical$(झवेरभाई पटेल, चौथी संतान)

4. **Arguments of a single predicate**: If two actor mentions are arguments of a single verbal predicate (except copula or copula-like verbs), then it is unlikely that these mentions are coreferences of each other. In other words, such arguments represent two distinct *semantic roles* of a single verbal predicate. "Semantic Role Labelling" (SRL) is itself a difficult problem and there are no annotated SRL datasets for Hindi. Hence, to find arguments of verbs, we use dependency parsing as a surrogate for full-fledged SRL. An actor mention is said to be an argument of a verb if the verb is ancestor of the actor mention in dependency tree and there are no other actor mentions on the path to the verb. E.g., Figure 1 shows dependency tree for the sentence सरदार पटेल ने उन्हे कर न देने के लिये प्रेरित किया। *(Sardar Patel inspired them not to pay the taxes.)* Here, for the verbal predicate किया, सरदार पटेल (*agent*) and उन्हे (*patient*) are its arguments. For all pairs for such arguments of a single verb, we add a soft rule in the MLN indicating that the actor mentions in the pair do not refer to a single real life actor.

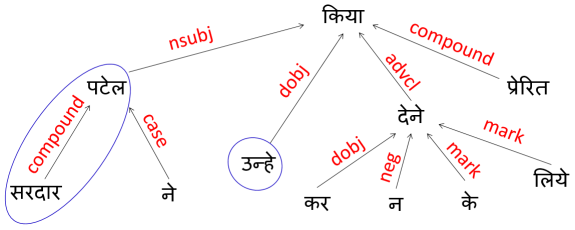


Figure 1: Dependency tree for the example sentence

$$10.0\ NonIdentical(\text{सरदार पटेल}, \text{उन्हे})$$

Antecedents: For each pronoun, we identify a certain "antecedent" nominal actor mention. The antecedent mention may precede the pronoun in the current sentence or be present in the previous sentence. To ensure that there is only one antecedent (y) for each pronoun (x), we add following rule with infinite weight:

$$IsAntecedent(x, y) \wedge (y \neq z) \Rightarrow \neg IsAntecedent(x, z).$$

Also, the pronoun (x) is likely to be coreferent of this antecedent (y). We incorporate this information with the following rule. The rule is a soft rule because it represents a weak assumption.

$$5.0\ IsAntecedent(x, y) \wedge PronounLike(x) \wedge \neg PronounLike(y) \Rightarrow Coref(x, y)$$

Consider following text fragment from a narrative: किसान अंग्रेज सरकार से कर में छूट की मांग कर रहे थे। तब वे सरदार पटेल से मिले। *(The farmers were demanding a discount in the taxes from the British government. During that time they met Sardar Patel.)*

Here, for the pronoun वे *(they)*, we consider किसान *(farmers)* as its candidate antecedent which is its closest preceding and type-compatible nominal actor mention. Hence, for the given example, we add following evidence to the MLN:

$$IsAntecedent(\text{वे}, \text{किसान})$$

It is important to note here that even though अंग्रेज सरकार *(British government)* is the closest nominal actor mention for वे *(they)*, it gets skipped on grounds of type incompatibility (ORG vs PERSON). Also, this is an inter-sentence rule which enables us to establish inter-sentence coreference links.

In addition to pronouns, we identify antecedents for certain nominal actor mentions which are similar to "definite" mentions in English. These nominal actor mentions are generally preceded by a demonstrative pronoun. E.g., केवल तीन रियासतें छोड़कर उस लौह पुरुष ने सभी रियासतों को भारत में मिला दिया *(Leaving only three states, that Iron Man was able to merge all others into India.)* In this example sentence, लौह पुरुष *(Iron Man)* is a definite mention because it is preceded by a demonstrative pronoun उस *(that)*. Hence, we find other antecedent nominal mentions for लौह पुरुष, one of which is likely to be its coreferent.

3.3 Inference

First, an MLN is created for a given narrative using the above-mentioned rules and evidences. Then, we run marginal inference in this MLN for the query predicate $Coref(x, y)$. We select actor mention pairs (x, y) for which probability of $Coref(x, y)$ is above a certain threshold. These pairs represent coreference links. We further add additional links so as to ensure transitivity of coreferences and get final coreference clusters.

4 Experimentation Details

In this section, we explain the datasets, ground truth creation, the experimental setup, evaluation methodology and results with their analysis.

4.1 Datasets

We select four narratives from Hindi Wikipedia each corresponding to an important event or person in India's history and employ them as the datasets for our experiments. Table 3 describes basic statistics about the four datasets.

The raw Wikipedia text contained multiple issues which we corrected by performing some basic cleaning. The steps involved are as follows.

- Spelling correction: We observed multiple instances of incorrect spellings for words which we corrected manually by hand. For instance, the word नबाव *(nabaw)* was used a few times in place of intended word नवाब *(nawab)*.

- Spelling normalization: For difficult names like सिराजुदौला *(Sirajudaulah)* a single spelling was fixed and its multiple variations were normalized to the chosen canonical one. We also did this for names occurring with incorrect spellings. For example, we replaced the incorrect पी. वी. मेनन *(P. V. Menon)* with the correct utterance वी. पी. मेनन *(V. P. Menon)*

- Sentence splitting: The sentence end marker in Hindi is the purn viram sign । which is

Dataset	#Sentences	#Words	#Actor Mentions
sardar[3]	90	1459	305
plassey[4]	70	1214	243
shivaji[5]	70	1224	256
emergency[6]	56	1221	197

Table 3: Dataset details[7]

different from the English full stop (.) making it an unambiguous sentence end marker. Sentence splitting can thus be performed easily by splitting on "। " i.e. the purn viram followed by a space. However, at multiple places in the Wikipedia text sentences were either ending abruptly without an end marker or the next sentence started right behind the marker without any space. These cases were handled by splitting sentences manually.

- Wiki meta-data removal: The narratives were obtained directly from the Wikipedia articles available on the web and hence, Wikipedia meta-data such as reference numbers, bullets, inline links, etc. were present. These unwanted characters were also removed.

Apart from the four Hindi Wikipedia based datasets, we use another dataset IIITH Hindi Coreference (Dakwale, 2014; Mujadia, 2017) dataset. Unlike the earlier four datasets, coreference annotations were available for IIITH dataset. However, we had to manually revise the annotations due to the following reasons:

- The dataset contains annotations for non-actor mentions as well. E.g., फिल्म महोत्सव में प्रकाश झा की फिल्म अपहरण का भी प्रीमियर होना है। इस फिल्म में बिपाशा बसु ने भी बतौर अभिनेत्री काम किया है। *(In the film festival, Prakash Jha's film Apaharan also has its premier. Bipasha Basu has acted in the movie as a lead actress.)* Hence, we discarded the non-actor mentions like फिल्म महोत्सव *(film festival)*, फिल्म *(film)* and अपहरण *(Apaharan)*.

- The dataset did not annotate entity types (PERSON, ORGANIZATION or LOCATION) for the mentions. Hence, we added actor types for all the actor mentions.

We manually revised annotations for 10 news articles out of the 275 news articles contained in the original dataset. These 10 new articles amounted to 156 sentences, 3412 words and 463 actor mentions.

4.2 Developing Ground Truth

An important part in the development of the ground truth is identification of actor mentions of the three types: named entities, generic NPs and pronouns for each dataset. The following tagging guidelines were set for guiding the actor mention identification.

- Tag all named entities occurring as separate noun phrases. For example, The phrase महात्मा गांधी *(Mahatma Gandhi)* needs to be tagged in महात्मा गांधी के विचारों से वे प्रेरित हुए । *(He was inspired by Mahatma Gandhi's thoughts.)*

- Tag all named entities occurring as part of a noun phrase even if the whole noun phrase is not a PERSON, LOCATION or ORGANIZATION. For example, बारडोली *(Bardoli)* needs to be tagged in बारडोली सत्याग्रह में पटेल का महत्वपूर्ण योगदान रहा। *(Patel had a pivotal role in the Bardoli Satyagraha.)*

- Tag all generic NPs and pronouns which refer to PERSONs, LOCATIONs and ORGANIZATIONs.

- Tagging of certain generic NPs needs to be carried out depending on the context they occur in and the noun they modify. For example, in the sentence उप-प्रधानमंत्री सरदार पटेल ने भारत को जोडने का कठिन परिश्रम किया । *(Deputy Prime Minister Sardar Patel performed the difficult hardwork for uniting India.)*, the generic phrase उप-प्रधानमंत्री *(Deputy Prime Minister)* behaves as an adjectival modifier to सरदार पटेल *(Sardar Patel)* and hence, the whole phrase उप-प्रधानमंत्री सरदार पटेल *(Deputy Prime Minister Sardar Patel)* gets marked as a single mention. However in the sentence भारत के उप-प्रधानमंत्री सरदार पटेल ने रियासत विभाग का गठन कीया । *(India's Deputy Prime Minister, Sardar Patel constituted the States Ministry.)*, the generic phrase उप-प्रधानमंत्री would be tagged separately as it is

[3]https://hi.wikipedia.org/wiki/वल्लभ_भाई_पटेल
[4]https://hi.wikipedia.org/wiki/प्लासी_का_पहला_युद्ध
[5]https://hi.wikipedia.org/wiki/शिवाजी
[6]https://hi.wikipedia.org/wiki/आपातकाल_(भारत)
[7]The datasets and ground truth will be made available if the paper gets accepted.

being modified by the phrase भारत के *(India's)*. So, four segments should be marked from the sentence namely भारत *(India)*, उप-प्रधानमंत्री *(Deputy Prime Minister)*, सरदार पटेल *(Sardar Patel)* and रियासत विभाग *(States Ministry)*.

Another important part of the ground truth is annotating a canonical mention for each actor mention to which it resolves to. Each canonical mention represents a cluster of actor mentions in which each mention is a coreference of other mentions. Also along with each actor mention, the type of the actor (PERSON, LOCATION or ORGANIZATION) is also specified by the annotator.

Four annotators were employed for creation of ground truth data. Each annotator tagged one dataset and then cross verified it with one other annotator. Tricky cases were discussed and resolved unanimously making sure the tagging guidelines were met and all annotators agree.

4.3 Experimental Setup

To tune the set of rules and their weights in the MLNs, we use the `sardar` dataset as a development dataset. The rest three datasets are only used for experimentation and reporting results. Also, as the main aim of the paper is to perform coreference resolution, we use the mentions identified in the text as a part of the ground truth as a starting point and run the MLN based algorithm to resolve these gold mentions.

To capture linguistic knowledge for predicates of the MLN we need the dependency parse of Hindi sentences. We used the Parsey Universal parser[8], available as part of the Google's Syntaxnet toolkit, which is a state-of-the-art open source dependency parser for 40 different languages.

For the implementation of the MLN, we use the open source MLN inference engine known as tuffy[9] (Niu et al., 2011). It supports marginal and MLE based inference. We use marginal inference in tuffy implemented through the MC-SAT algorithm with number of samples as 100. As this is an approximate inference, we run the inference for each narrative five times and report results averaged over the five runs.

[8] https://github.com/tensorflow/models/blob/master/research/syntaxnet/g3doc/universal.md

[9] http://i.stanford.edu/hazy/tuffy/

Dataset	Setting	MUC	B^3	CEAFe	Avg
sardar	B1	74.63	59.33	50.57	61.51
	B2	70.55	69.67	63.57	67.93
	MLN	73.35	71.98	66.05	**70.46**
plassey	B1	72.3	53.03	47.93	57.75
	B2	62.36	61.39	62.74	62.16
	MLN	68.09	63.33	63.31	**64.91**
shivaji	B1	71.92	57.01	55.50	61.48
	B2	70.96	70.20	69.02	**70.06**
	MLN	71.07	70.00	65.88	68.98
emergency	B1	70.14	46.25	45.17	53.85
	B2	62.52	62.95	61.35	62.27
	MLN	62.93	63.62	62.83	**63.12**
IIIT-H	B1	67.53	53.51	42.84	54.62
	B2	59.24	50.42	38.81	49.49
	MLN	64.25	55.88	45.00	**55.04**

Table 4: F1 measures according to various metrics

4.4 Baselines and Evaluation

We developed following baseline approaches for Coreference Resolution of actor mentions:

1. **B1**: Here, a pair of actor mentions are said to be coreferences of each other if: i) they are lexically similar or ii) one of the mentions is pronoun and other is its type-compatible closest antecedent. Final coreference clusters are obtained by getting a transitive closure of such corefering pairs.

2. **B2**: This baseline uses the linguistic rules proposed by Patil et al. (2018). None of these rules capture any inter-sentence relation among actor mentions.

We evaluated the output of our MLN based system using three metrics widely used in the literature to report coreference resolution results. They are the MUC (Vilain et al., 1995), the B^3 (Bagga and Baldwin, 1998) and the CEAFe (Luo, 2005) metrics.

Table 4.4 reports the results obtained on the four datasets. It is important to note here that the approach proposed is unsupervised. Hence, the obtained accuracies are encouraging and entail further exploration.

4.5 Analysis

In Table 4, we report the comparative performance of our approach (MLN) with baselines. Our approach outperforms baseline B1 on all datasets and baseline B2 (Patil et al., 2018) on 4 out of 5 datasets. Even though baseline B2 is also based on MLN, our approach uses a richer set of rules such as:

- Using the antecedent rule which enables us to link inter-sentential coreferences to a moderate extent. However, baseline B2 only uses intra-sentence rules.

- Finding antecedents not only for pronouns but also for definite mentions

- Using SRL-like predicate-arguments for identifying non-identical actor mentions

In general, English coreference resolution techniques exploit gender and number compatibility in addition to entity type compatibility. However, in Hindi, we observed that gender and number compatibility do not hold in large number of cases. As an example, the pronoun वे *(they / he)* may refer to the singular mention सरदार पटेल *(Sardar Patel)* or the plural mention किसानों *(farmers)* depending on the context. This is an example of a Hindi-specific phenomenon of using plurals to indicate respect (आदरार्थी बहुवचन). Similarly, the pronoun उन्हें *(him / her)* may refer to either the masculine mention संजय गांधी *(Sanjay Gandhi)* or the feminine mention इंदिरा गांधी *(Indira Gandhi)* depending on the context.

We also observed that considering only the nearest nominal actor mention as the antecedent, is not always correct. For example, in the sentence लालू प्रसाद यादव के साले सुभाष यादव ने उन पर गलत टिकट वितरण का आरोप लगाया *(Lalu Prasad Yadav's brother-in-law Subash Yadav accused him of incorrect candidacy allocation.)*, the pronoun उन *(him / her)* actually refers to लालू प्रसाद यादव *(Lalu Prasad Yadav)* and not सुभाष यादव *(Subash Yadav)* which is the closest type-compatible actor mention. This requires further exploration on using multiple preceding actor mentions as antecedents.

5 Conclusion and Future Work

In this paper, we focussed on resolving coreferences of actor mentions in Hindi narrative text. The proposed approach is linguistically grounded and uses Markov Logic Networks (MLN). The MLN framework proved to effective in representing various pieces of information characterizing linguistic knowledge relevant to coreference resolution. Unlike neural or other supervised approaches, our approach does not need a large amount of coreference annotated data. We also contributed four new datasets annotated with actor mentions and their coreferences. Our approach outperformed two baselines including a strong recent one developed for English narrative text.

In future, we plan to build an end-to-end system which first identifies actor mentions and then resolves coreferences among them. We also intend to analyze the robustness of our rules in this scenario of using predicted actor mentions.

References

S. Agarwal, M. Srivastava, P. Agarwal, and R. Sanyal. 2007. Anaphora resolution in hindi documents. In *2007 International Conference on Natural Language Processing and Knowledge Engineering*, pages 452–458.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*, volume 1, pages 563–566. Granada.

Kevin Clark and Christopher D. Manning. 2016. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 643–653. Association for Computational Linguistics.

Praveen Dakwale. 2014. *Anaphora Resolution in Hindi*. Master's thesis, Language Technologies Research Center (LTRC), International Institute of Information Technology, Hyderabad - 500 032, INDIA.

Pedro M. Domingos and Daniel Lowd. 2009. *Markov Logic: An Interface Layer for Artificial Intelligence*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Kamlesh Dutta, Nupur Prakash, and Saroj Kaushik. 2008. Resolving Pronominal Anaphora in Hindi using Hobbs's algorithm. In *Web Journal of Formal Computation and Cognitive Linguistics*.

Barbara J Grosz, Scott Weinstein, and Aravind K Joshi. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational linguistics*, 21(2):203–225.

J Hobbs. 1986. Readings in natural language processing. chapter Resolving Pronoun References, pages 339–352. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197. Association for Computational Linguistics.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *HLT/EMNLP 2005*, pages 25–32. Association for Computational Linguistics.

Vandan Mujadia. 2017. *Capturing and Resolving Entities and their Mentions in Discourse*. Master's thesis, Language Technologies Research Center (LTRC), International Institute of Information Technology, Hyderabad - 500 032, INDIA.

Vincent Ng. 2017. Machine learning for entity coreference resolution: A retrospective look at two decades of research. In *AAAI, 2017*, pages 4877–4884.

Feng Niu, Christopher Ré, AnHai Doan, and Jude W. Shavlik. 2011. Tuffy: Scaling up Statistical Inference in Markov Logic Networks using an RDBMS. *PVLDB*, 4(6).

Sangameshwar Patil, Sachin Pawar, Swapnil Hingmire, Girish Palshikar, Vasudeva Varma, and Pushpak Bhattacharyya. 2018. Identification of alias links among participants in narratives. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 63–68.

Rashmi Prasad and Michael Strube. 2000. Discourse salience and pronoun resolution in hindi. *Penn Working Papers in Linguistics*, 6(3):189–208.

Bhargav Uppalapu and Dipti Misra Sharma. 2009. Pronoun resolution for hindi. In *7th Discourse Anaphora and Anaphor Resolution Colloquium*.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th conference on Message understanding*, pages 45–52. Association for Computational Linguistics.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 57.

Deep Learning methods for Semantic Role Labeling in Indian Languages

Aishwary Gupta, Akshay Pawale and Manish Shrivastava
IIIT Hyderabad, India
{aishwary.gupta, akshay.pawale}@research.iiit.ac.in
m.shrivastava@iiit.ac.in

Abstract

We introduce a supervised deep learning model for Indian languages namely, Hindi and Urdu which uses minimal syntax and yet improves over the current baseline for these languages significantly. We progress with three different models inspired from the recent advancements in this field. In the first model we make use of sequence modeling to generate dependency path embeddings and jointly learning the classification process i.e., Identification and Labeling of arguments. The second model is a syntax-agnostic model where we encode the full sentence using a bi-directional LSTM encoder only using the raw words/tokens. The third and the final model adds dependency label to the previous model making it syntax-aware and performs very well compared to the other models. Finally, we talk about evaluation metrics and analysis of these models.

1 Introduction

Semantic role labeling (SRL) is one of the fundamental tasks in Natural Language Processing (NLP) which aims to automatically learn about the predicate-argument structure for each predicate in given a sentence as the input. It is a type of shallow semantic parsing which labels all the involved constituents in a sentence pertaining to each verb (predicate), answering queries such as who did what to whom, when, and where etc. Semantic roles for a predicate generally are Agent, Patient etc., and adjuncts like Temporal, Locative, Cause etc. Since SRL provides a semantic analysis of a sentence, it can be used in many fields of NLP. Applications of SRL have been shown in topics like information extraction (Bastianelli et al., 2013; Christensen et al., 2010), question-answering (Pizzato and Mollá, 2008; Shen and Lapata, 2007) and machine translation (Liu and Gildea, 2010). SRL has been a trending topic in the research areas of NLP and as a result we have seen great contributions in the form of systems and datasets for various languages. Though much work has been done towards SRL for resource-rich languages like English and Chinese but SRL for Indian Languages (ILs) was pioneered quite recently (Nomani and Sharma, 2016) . Following which another system was recently published (Gupta and Shrivastava, 2018). Both these systems are based on traditional approaches towards SRL as seen in Gildea and Jurafsky (2002) and Xue and Palmer (2004). In this work, we try to apply relatively recent approaches for SRL which include the use of neural networks and find out which works to be the best. Formally, we build three systems-
Model A: Encoding paths between constituents and the predicate using LSTM
Model B: Encoding the whole sentence in a bidirectional LSTM without using any syntax at all.
Model C: Adding a syntactic feature to Model B becoming syntax-aware.

Model A is an extension for the traditional approach where we, (1) first perform Argument Identification i.e., binary classification and (2) then Argument Classification on the probable arguments passed on by step 1. This is often done by training classifiers like SVM on features extracted from the training data. The features are based on syntactic information which is crucial for this approach (Punyakanok et al., 2008). In this model, we find an embedding for the dependency path between a constituent and the current predicate and combine that with other features as used

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 59–68

in earlier systems (Roth and Lapata, 2016). This model performs slightly better than the current baseline in ILs (Gupta and Shrivastava, 2018). The gold data is a human-annotated corpus with fully descriptive dependency trees, it has a few chances of error. But in reality we would be given a raw input sentence that would then be parsed by the system and could bring in more error than gold data and affect training badly. Results using automatic parses in the first work (Nomani and Sharma, 2016) verify this. So we apply the recent advancements in SRL which are syntax-agnostic. Model B takes the input as a raw sentence in form of tokens and the only prior information is what are the predicate(s) in this sentence. The sentence is considered as a sequence and we perform SRL as a sequence labeling task. Surprisingly, even for such low resourced language as Hindi and Urdu are, this outperforms Model A and hence the previous baseline. Model C, along with the words adds the dependency label as a feature in the sequence and consequently performs even better than Model B which can be attributed to the use of syntactic feature(s). This makes it the best system available for Hindi and Urdu.

2 Related Work

There are mainly two ways to approach semantic role labeling. The first approach is a traditional one which uses some features extracted out of the gold data. The gold data is a syntactic parsed data with details such as constituent boundaries, syntactic categories, the whole tree structure of the sentence with syntactic relations, part-of-speech(POS) tags for each token in the sentence etc. These features are then used to train a linear classifier like SVM for the tasks of Argument Identification and Argument Classification subsequently (Xue and Palmer, 2004). These tasks can also be done as a single multi-label classification task (Surdeanu and Turmo, 2005). The best predicate-argument structure is chosen at inference stage by techniques like integer linear programming (Punyakanok et al., 2004) or dynamic programming (Täckström et al., 2015). At this stage, structural constraints may lead to improvement in results (Punyakanok et al., 2008). These constraints are either linguistically driven or they exist as a result of the annotation process.

In the last few years, significant work has been done towards fully syntax-agnostic approaches using deep neural networks. Collobert et al. (2011) was the pioneer in introducing such an approach which considers SRL as a sequence labeling task using a convolutional neural network. It takes as input the raw sentence and the constituent boundaries. Although, their approach could not beat the best systems which were still using traditional approaches. The breakthrough in syntax-agnostic SRL was done by Zhou and Xu (2015) whose system differs from Collobert et al. (2011) by using a Deep Bidirectional LSTM network which takes only the predicates indices as input along with the raw sentence. More recently, an end-to-end semantic role labeler was built by He et al. (2017) in which they first identify the predicate(s) in a given input sentence and then for each identified predicate, label the arguments with respective boundaries in the sentence. In such models, the raw sentences tokens are first converted to vector form embeddings taken from pre-trained models. All the above was done for span-based SRL. Marcheggiani et al. (2017) did a similar approach for dependency based SRL. In dependency based SRL, the task it to label the head words of constituents given the gold data has a dependency tree structure. The data for both Hindi and Urdu uniquely have a dependency parsed structure with span-based labeling unlike other languages like English, Chinese etc. where span-based SRL is done for syntactic parses whilst head/dependency-based SRL is done for dependency parses. Before these systems became popular, neural networks were used in syntax-aware systems also (FitzGerald et al., 2015; Täckström et al., 2015; Roth and Lapata, 2016).

The first one to work on SRL for ILs was Nomani and Sharma (2016). They use simple features like dependency labels, syntactic categories, head word of the chunk, head words POS tag, Named entities etc. Gupta and Shrivastava (2018) improved the labeler by introducing features like word embeddings to address the data sparsity issue, path from chunk[1] to predicate and post-positionals of the chunk. These fall under the traditional approaches discussed above. Our first model uses neural method but only to model the dependency path and therefore it still majorly relies on syntactic features. Vaidya et al. (2011) has shown that the predicate-argument structure is

[1]Similar to previous work, we call a word-span/constituent as a chunk.

closely related to dependency relations. The same was proven in the system by Nomani and Sharma (2016) where dependency labels used alone gave good F1-scores for both Hindi and Urdu. Though when they used automatic labels, there is a huge drop in results mainly due to errors in the dependency parse. Dependency/Syntactic parsing in itself is a difficult problem and hence we build a fully syntax-agnostic model to eliminate our reliance on syntax. Although, our third model shows that syntax can still improve performance if it is free from errors.

3 Models

3.1 General Pipeline

The general architecture of SRL is explained in this section. The first step is to identify semantic predicates in the input sentence. In English propbank, there are both verbal and nominal predicates (Hajič et al., 2009), whilst in Hindi and Urdu only verbal predicates are present as of now. Next, the system should disambiguate word-sense of the predicate in consideration. Propbank has multiple senses for verbs and each sense of the same verb can have different labels. This step can be used to improve training or it can just be used at inference time by looking in the corresponding frameset files, the specific roles a verb with given sense can have. The next two steps are Argument Identification and Argument Classification. Argument Identification is done because a high number of candidate arguments have the role **NULL** which may affect the decision boundaries if a classifier is learned directly on full candidates (Xue and Palmer, 2004). Argument Classification is then done to label the remaining candidates which are the most potential arguments from the previous step. For Hindi and Urdu, labeling is done at the chunk/constituent level. A re-ranker using integer linear programming or dynamic programming can be applied as a last step to get the best argument structure for the sentence. For each predicate, we have the identified arguments, each with its scores for every role class. Now we may apply some constraints(structural/linguistic) on the possible output structure and penalize some of the outcomes. Finally we get the result with the best possible predicate-argument structure.

3.2 Sequence Modeling and LSTM

The long short-term memory (LSTM) network is an alternative architecture for a Recurrent neural network (RNN) where each block is a LSTM cell/unit instead of a typical neuron. RNNs process a sequence token by token where each block is given the previous information plus the current token information.

x and y are the input and output respectively, (t) denotes the time step, w_f^m and w_f^m are the matrix from input or recurrent layer to the hidden layer and σ is the activation function. Without $y^{(t-1)}$ term, the RNN model becomes a feed forward network only with number of layers equal to sequence length. RNN is shown in the equation below:

$$y_m^{(t)} = \sigma(\sum_f w_f^m x_f^{(t)} + \sum_i w_i^m y_i^{(t-1)}) \quad (1)$$

When we take one-hot encoded/binary-coded features(as vector), the representation is not effective since data for Hindi and Urdu is quite sparse and vector size is large. To address this, we experiment with recurrent networks to improve the feature representation/encoding and reduce its dimensionality. We use RNNs variant, the LSTM network because it is known to handle long range dependencies (Zhou and Xu, 2015) and in a sentence, the current word is quite dependent on distant words. Also, gradient parameters may vanish or explode especially in processing long sequences (Bengio et al., 1994). To resolve these issues, long short-term memory (Hochreiter and Schmidhuber, 1997) was presented.

LSTM Network. Long short-term memory(LSTM) (Hochreiter and Schmidhuber, 1997; Graves et al., 2009) has a modified architecture with respect to a simple RNN. Memory blocks are used instead of the hidden neural units. The memory block may contain several cells which are activated three multiplicative gates: the input gate, the forget gate and the output gate. These changes improve the RNN model for sequence modeling. Figure 1 shows a basic LSTM cell.

y is the output from memory block. h is the hidden value which equals $'y'$ from the basic RNN model discussed above. c is the cells state value for block m. Number of cells in a block is fixed to be one. α, β and γ stand for the input, forget and output gates activation values. All three mul-

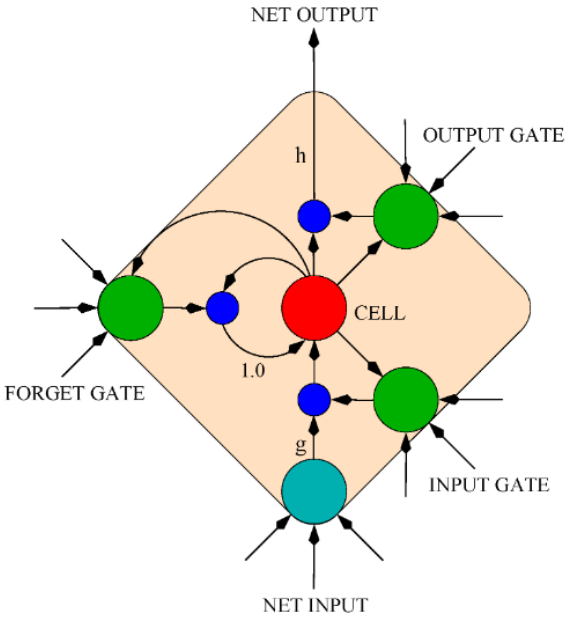


Figure 1: LSTM memory block with one cell. (Graves et al., 2009)

tiplicative gates have different activation σ respectively and the computations are done as follows:

$$
\begin{aligned}
h_m^{(t)} &= \sigma_h(\sum_f w_{f,h}^m x_f^{(t)} + \sum_i w_{i,h}^m y_i^{(t-1)}) \\
\alpha_m^{(t)} &= \sigma_\alpha(\sum_f w_{f,\alpha}^m x_f^{(t)} + \sum_i w_{i,\alpha}^m y_i^{(t-1)} + w_\alpha^m c_m^{(t-1)}) \\
\beta_m^{(t)} &= \sigma_\beta(\sum_f w_{f,\beta}^m x_f^{(t)} + \sum_i w_{i,\beta}^m y_i^{(t-1)} + w_\beta^m c_m^{(t-1)}) \\
\gamma_m^{(t)} &= \sigma_\gamma(\sum_f w_{f,\gamma}^m x_f^{(t)} + \sum_i w_{i,\gamma}^m y_i^{(t-1)} + w_\gamma^m c_m^{(t)})
\end{aligned}
\tag{2}
$$

The gates allow the cells to store and access information over long periods of time/long steps. When the input gate is closed, the new coming input information will not affect the previous cell state. Forget gates remove some historical information over time steps. The output gate should be open for a cell, if rest of the network has to access this cells stored value. In NLP related problems, structural knowledge can be accessed by training the sequences both forward and backward so that the contextual information from left as well as right can be incorporated for better inference. Thus bi-directional LSTM (BiLSTM) was proposed (Schuster and Paliwal, 1997). The BiLSTM which we use is slightly different from their's. We take a LSTM layer to processes the sequence in forward direction whose output is taken by the next LSTM layer as input, where the connections are in backward direction. Pairs of these forward and backward layers can be stacked together to make a Deep BiLSTM proposed in earlier work (Zhou and Xu, 2015).

3.3 Model A - Path embedding model

This model closely follows the architecture of PathLSTM (Roth and Lapata, 2016) and is shown in Figure 2. Given a candidate chunk, first we find the path from this chunk to the predicate being considered and initialize its embedding as follows. Each node in the path is represented as the head-word embedding, POS tag of the chunk, dependency relation with the parent chunk, all three concatenated. Note that this path makes connections at the chunk level only and ends at the predicates chunk. The head-word embeddings are pre-trained embeddings computed similarly as previous work (Gupta and Shrivastava, 2018). The POS tag and dependency relation are given a one-hot vector initialization. Now, we use our sequence model to compute the vector representation of this path.

This differs from the previous work (Gupta and Shrivastava, 2018) as they represent each distinct path as a one-hot encoded vector which is probably not a very optimal way since the number of distinct paths is quite high. For example, in Hindi propbank, even considering only the chunk POS categories[2] in our dependency paths from argument to predicate, nearly 2400 unique paths are present in the training data. If we take the dependency labels only in path (considering direction as argument to predicate), there are around 5900 distinct paths out of which major paths are k1↑root, k2↑root, ccof↑root etc., which occur for only 3.7%, 2.7% and 2.4% respectively (root signifies the predicate). This implies, for almost all paths amongst these, the training data is very less to make any significant improvement in learning from path as a feature. Further, using one-hot implies that we assume that certain paths are likely to impact role labeling in a similar way which may not be true (Roth and Lapata, 2016). Thus, some representation learning should be done, instead of taking the full path as it is.

The dependency path is given to the LSTM

[2]Number of distinct POS categories at chunk level is 12.

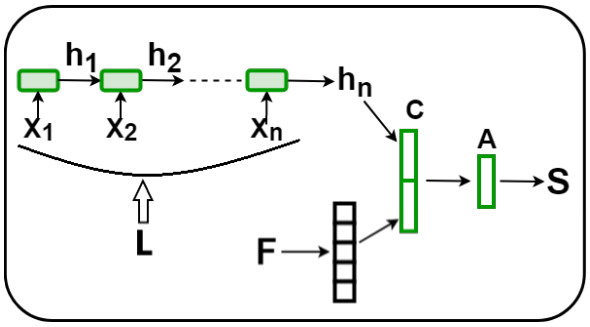


Figure 2: Model A - Path embeddings with LSTM

network as a sequence. The path is taken from the argument chunk to the predicate chunk. Particularly, an element x_i corresponds to the head-word of the chunk w_i, followed by POS category of chunk and then its dependency relation with the next chunk in path, x_{i+1}. The last blocks output state from the network gives us the embedding of this path. As shown in Figure 2, the embedding of a dependency path, specifically is h_n which is returned by the LSTM layers last block after the input of the last element of the sequence, x_n, which corresponds to the initialized encoding of the predicate's chunk.

We use syntactic categories, dependency roles and head-word of chunks in the path because all of them can affect the decision of role labeling (Roth and Lapata, 2016; Gupta and Shrivastava, 2018).

Model A is depicted in Figure 2 and its components are: (1) **L** is the LSTM network which takes input of the length of our path where each node is initialized as discussed above, (2) **F** is the additional input layer which takes features other than the path as input, (3) layer **C** concatenates two neural layers: the upper block is a linear neural layer which takes last node (h_n) of L (dependency path embedding) as input and the lower block is another linear neural layer which takes input from F, (4) layer **A** applies an activation function on the input it receives, and finally (5) **S** is a softmax classifier which produces output for each class $k \in K$ depending on the task(identification or classification). This makes a joint learning model which learns dependency path embeddings and performs SRL as well. Formally, we are given a initialized dependency path X with elements $x_i \in \{x_1, ..., x_n\}$ where n is the length of the path and the features F as one-hot encodings. The LSTM formalization computes hidden embeddings at each step $h_i \in \{h_1, ..., h_n\}$ but we only need the embedding h_n which makes our LSTM network slightly modified than the usual one because gradient parameters will be updated depending on just the last blocks output. We formalize the next layers as follows :

$$C = (W_L h_n + b_L)|(W_F F + b_F)$$

$$A = relu(C)$$

$$S_k = A_k / \sum_{k \in K} A_k$$

We perform the training for argument identification and argument classification separately following the findings from earlier work in English (Xue and Palmer, 2004) as well as for Hindi and Urdu (Nomani and Sharma, 2016). This also means that different path embeddings are learned depending on what the task is. The features F are taken as it is from the current baseline (Gupta and Shrivastava, 2018) for ILs for making our comparison more obvious. These features are - predicate word(verb) root form, its suffix separately, head word of the candidate chunk taken from pre-trained embeddings, candidate chunks vibhakti(post-positional), head word POS tag, candidate chunks POS category.

3.4 Model B - Syntax-agnostic deep model

This model takes the sentence as a sequence processed word by word. A sequence say of length L is processed n_p times if the number of predicates in the sentence is n_p. Hence, the time complexity of this model is $O(n_p L)$. At each step in the sequence, the current words embedding and a binary bit indicating whether word is itself the predicate or not, is given as the input. These are the only features needed to train our network.

Given a sentence-predicate pair (s, v) as the input, we have to predict the output sequence y. We have used IOB tagging (Collobert et al., 2011; Zhou and Xu, 2015) for this problem. Therefore, each $y_i \in y$ should belong to the set of IOB tags. The set contains the tags, O - is given to words outside the argument chunk, B_k - is given to words at beginning of the chunks and I_k - is given to the words inside the chunks. k denotes the various roles shown in section 4. Let the length of the sequence be n, where $n = |s| = |y|$. Our goal is to find the highest scoring tag sequence y from all the possible tag sequence for an input. Our model uses a Bidirectional LSTM (BiLSTM) network to

learn a locally decomposed scoring function determined by the input: $\sum_{t=1}^{n} \textbf{log}p(y_t|s)$. To incorporate constraints like IOB order, structural constraints(explained later in this section), we extend the scoring function (He et al., 2017) with penalization terms:

$$f(s,y) = \sum_{t=1}^{n} logp(y_t|s) - \sum_{c \in C} c(s, y_{1:t})$$

Given the input s and length-t prefix $y_{1:t}$, each constraint function c applies a non-negative penalty on the scoring function f.

The model is depicted in Figure 3. The input colored as red is the raw word, binary bit b_i to indicate whether word w_i is the predicate. The dotted box next to it is shown to incorporate additional features if required, which is basically our third model and not of use in this model. The input is then embedded as the concatenation of word embedding and the binary bit at the embedding layer. The next layer is the beginning of our BiLSTM network. Layer L_k is forward if k is odd and backward if k is even. We chose the number of layers as two for reasons given in section 6. Recent best works (Zhou and Xu, 2015; Marcheggiani et al., 2017; He et al., 2017) in English SRL have used up till 8 layers. Finally, the output of this goes to a softmax layer to compute a locally normalized distribution over the output tags.

Constrained Decoding. The approach above does not yet apply any constraints on the output structure. We use A search over tag prefixes to apply constraints on the output structure at decoding time (He et al., 2017). We list some example constraints as follows:

IOB Constraints: We need to ensure that our system does not produce invalid IOB tagging such as B_k tag followed by I_k tag or say I-ARG0 followed by B-ARG1 etc. We apply infinitely high penalty for such transitions.

SRL Constraints: Though there have been no constraints described in the data or in previous work for Hindi and Urdu propbanks, but some structural constraints have been applied for English SRL (Punyakanok et al., 2008; Täckström et al., 2015). We only experiment with the Unique core roles constraint, i.e., numbered arguments (ARG-0,1,2,3) can occur at most once for each predicate.

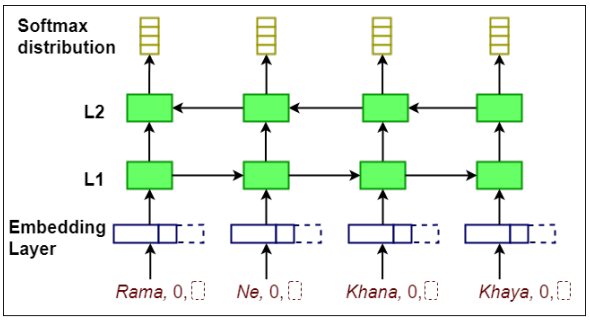


Figure 3: Model B and C - 2 Layer BiLSTM model.

3.5 Model C - Syntax-aware deep model

We build a third model to see the effect of adding syntax to Model B which is fully syntax agnostic. Going with the findings by Nomani and Sharma (2016) that dependency label alone gives great results, we go ahead with it as the only feature to be used. In Figure 3, the red colored input with dotted box is meant to incorporate any other features and this is where we add dependency label. Since the dependency parsing in Hindi and Urdu propbanks is only till the chunk level, so a whole chunk, i.e., all tokens inside are assigned the same dependency relation label.

Formally, at the embedding layer, we now concatenate embeddings of the word, the binary bit indicating if this word is the predicate and the dependency role encoded as a one-hot vector. After this, the training is exactly same as model B.

4 Dataset

We carry out our experiments on Hindi Propbank and Urdu Propbank exactly on the sections used in earlier work. We use the same train and test set for both the languages. Hindi and Urdu propbank are still in the process of annotation. These are build on top of the respective treebanks which has dependency parsed trees with parsing up till the chunk level. The semantic label annotation is also done at chunk level. Chunk boundaries, POS categories of tokens as well as the whole chunk, morphology features etc. are already annotated in the gold corpus. The treebanks and hence the propbanks (since it is build on treebank) of both languages are represented in the Shakti Standard Format (Bharati et al., 2007). Gupta and Shrivastava (2018) did a 5-fold cross validation on this data and the results showed a very slight change with

respect to the train-test split used earlier (Nomani and Sharma, 2016). This shows that the data distribution in the train-test splits is normalized and hence, we don't perform cross-validation on our models in this work. They have also given a good explanation about the data set but they missed to give out any statistics about the data. Therefore, we decided to provide full details on the data with this work and show them in Table 1. The exact data file names used for training and testing are listed in section 8 of this paper.

	Hindi train	Hindi test	Urdu train	Urdu test
Sentences	1643	448	4657	1234
Tokens	36690	9827	133058	35532
Final Sentences	1300	358	988	309
Final Tokens	30141	8285	33374	10628
Propositions	2309	631	1192	391
Verbs	166	94	40	24
Arguments	5872	1620	3745	1207
ARG0	1185	320	433	137
ARG1	2046	559	624	210
ARG2	175	33	99	27
ARG3	4	0	14	2
ARG2-ATR	352	77	53	8
ARG2-GOL	61	13	4	9
ARG2-LOC	54	11	137	54
ARG2-SOU	46	14	79	33
ARGM-LOC	679	210	399	136
ARGM-MNR	350	106	67	21
ARGM-TMP	328	97	210	72
ARGM-ADV	131	38	194	52
ARGM-PRP	114	31	102	26
ARGM-DIS	94	34	25	10
ARGM-EXT	92	23	10	4
ARGM-CAU	83	29	46	4
ARGM-MNS	42	13	24	7
ARGM-DIR	20	8	6	2
ARGM-NEG	7	2	13	1
ARGM-PRX	2	1	1194	393
ARGM-VLV	0	0	0	0
ARGM-MOD	2	0	1	0
ARGA	0	0	1	1
ARGC	0	0	0	0

Table 1: Hindi and Urdu Propbank data statistics.

Final Sentences and Final tokens signify the numbers after filtering out sentences with no predicate. The total number of distinct verbs in full Hindi and Urdu datasets after filtering are 186 and 46.

5 Experiments

Evaluation metrics. To compare Model A with the previous works, we have used the same evaluation metric as used by them, which is to give the results of the tasks of identification and classification separately. Since model B and C do not perform identification and classification separately, we cannot compare this directly to results of previous works and model A. We propose the use of the evaluation script used for Carreras and Màrquez (2005). It gives a more perceivable understanding of the results of a model since it compares the actual sentences, one-to-one from the gold corpus and from the predictions. According to the script, a prediction is counted correct if and only if it has the exact same chunk boundary and the same label. We also project the results of Model A to test data sentences and evaluate it with conll 2005 script to make a clear comparison of all models we built.

5.1 Model A

We train different networks for argument identification and classification using the PyTorch library for deep neural networks. For direct comparison with previous work, features other than path are same as the ones used in previous work (Gupta and Shrivastava, 2018) and evaluation metric is also the same. The hyper-parameters of the model are as follows:

Learning rate for identification is 0.001 and for classification is 0.0001. Dropout rate is 0 for both the tasks and hidden layer size of LSTM network is 100 for both the tasks. Layer C is composed of two neural blocks each of size 100, whose output of size 200 is sent over to layer A after concatenation. We have used Cross Entropy function to compute loss and Stochastic Gradient descent (SGD) for optimizations. The model was run for 200 iterations for identification task and 300 iterations for classification task. The results for Hindi and Urdu are given in comparison with their corresponding state-of-the-art in Table 2 and Table 3 for identification and classification respectively.

Language	Model	Precision	Recall	F1-score
Hindi	Gupta et. al	91.41	90.49	90.94
	Our Model	**93.35**	**93.29**	**93.32**
Urdu	**Gupta et. al**	**92.05**	**91.49**	**91.76**
	Our Model	91.50	91.17	91.33

Table 2: Argument Identification results.

Language	Model	Precision	Recall	F1-score
Hindi	Gupta et. al	65.04	66.62	65.80
	Our Model	**70.23**	**72.25**	**71.22**
Urdu	**Gupta et. al**	**86.72**	**86.37**	**86.54**
	Our Model	85.57	85.52	84.73

Table 3: Argument Classification results.

5.2 Model B and C

Our BiLSTM has only 2 layers - 1 forward LSTM and 1 backward LSTM. The hidden dimension is set to be 300. A softmax layer for predicts the output distribution. All weight matrices of the BiLSTM are initialized as random orthonormal matrices as described in Saxe et al. (2013). The pre-trained embeddings for both Hindi and Urdu are taken from Fast-Text (Bojanowski et al., 2017). Tokens that are not present in the pre-trained model are given a randomly initialized embedding. Size of the embedding is 300 for both languages.

Training: We use Adadelta (Zeiler, 2012) as optimizer with $\epsilon = 1e^6$ and $\rho = 0.95$ which are also the default parameters available in the proposed paper. We use mini-batches of size 50. We clip gradients with norm larger than 5 and set the RNN-dropout probability to 0.1. All the models are trained for 300 iterations without any early stopping since we dont have a development data to check the development loss. In the final model we only use the IOB constraints.

6 Results and Analysis

The problem with Model A is that it also depends on a feature template which can be language/data specific. Hindi propbank was created automatically while Urdu was purely human-annotated, this is why Urdu's dataset is better and thus it gives better results. The Urdu verbs are also very less in number as seen in Section 4 and thus the system doesn't have to learn a lot of predicates. Also, the majority arguments in Urdu belong to ARGM-PRX which is a complex noun-verb/adjective-verb predicate but this class achieves the best F-score and contributes heavily towards the results. This also means that the data sparsity is low.

In Model A, We chose the path configuration from argument chunk to predicate chunk because it gave better results than the reverse path which is also the actual dependency path. The first model learns the path embeddings and performs only slightly better than the previous work (Gupta and Shrivastava, 2018). The difference in this model and previous work is only that it uses LSTM to encode path while the previous work used a one-hot encoding for this feature. Though, in our case it provides only a slight gain. This can be attributed to the fact that the data available for each unique path is very less to learn a reliable embedding.

In Model B and C, number of layers is kept as 2 because increasing number of layers degraded performance. The primary reason for this could be attributed to less training points in data. We only use IOB constraints since they gave a significant improvement in results whilst the Unique Core Roles constraint did not. These models are overall the best performers as they are in a way learning more complex features from the whole sentence than the pre-defined features used in model A.

Language	Model	Precision	Recall	F1-score
Hindi	Gupta et al.	42.52	49.66	45.81
	Model A	44.38	50.53	47.26
	Model B	56.71	56.39	56.55
	Model C	**70.41**	**70.41**	**70.41**
Urdu	Gupta et al.	86.41	67.16	75.58
	Model A	85.01	66.91	74.88
	Model B	63.10	63.20	63.15
	Model C	**77.98**	**78.16**	**78.07**

Table 4: SRL full task results evaluated using conll-2005 shared task evaluation script.

7 Conclusion and Future Work

More data for both the languages can be helpful to improve the performance further and get a better analysis. A detailed report can then be achieved on what quantity of data is actually required for SRL in low-resource settings. Once more data is available, analysis of results will give the reasons why and how deep learning models perform better than the traditional ones. Results of our paper have shown that deep learning is performing very well even in such low data. This shows that more complex features have been missed when manually extracting features from a parsed sentence. Hence, better feature engineering also lies in the future scope of SRL for Indian Languages.

8 Data Sections

The names of the files in training set and testing set of Hindi are provided in the following link. https://github.com/ashg1910/indian_srl

References

E. Bastianelli, G. Castellucci, D. Croce, and R. Basili. Textual inference and meaning representation in human robot interaction. In *Proceedings of the Joint Symposium on Semantic Processing. Textual Inference and Structures in Corpora*, pages 65–69, 2013.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

A. Bharati, R. Sangal, and D. M. Sharma. Ssf: Shakti standard format guide. *Language Technologies Research Centre, International Institute of Information Technology, Hyderabad, India*, pages 1–25, 2007.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. ISSN 2307-387X.

X. Carreras and L. Màrquez. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the ninth conference on computational natural language learning*, pages 152–164. Association for Computational Linguistics, 2005.

J. Christensen, S. Soderland, O. Etzioni, et al. Semantic role labeling for open information extraction. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 52–60. Association for Computational Linguistics, 2010.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

N. FitzGerald, O. Täckström, K. Ganchev, and D. Das. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 960–970, 2015.

D. Gildea and D. Jurafsky. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288, 2002.

A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868, 2009.

A. Gupta and M. Shrivastava. Enhancing semantic role labeling in hindi and urdu. In K. Shirai, editor, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France, may 2018. European Language Resources Association (ELRA). ISBN 979-10-95546-24-5.

J. Hajič, M. Ciaramita, R. Johansson, D. Kawahara, M. A. Martí, L. Màrquez, A. Meyers, J. Nivre, S. Padó, J. Štěpánek, et al. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18. Association for Computational Linguistics, 2009.

L. He, K. Lee, M. Lewis, and L. Zettlemoyer. Deep semantic role labeling: What works and whats next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 473–483, 2017.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

D. Liu and D. Gildea. Semantic role features for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 716–724. Association for Computational Linguistics, 2010.

D. Marcheggiani, A. Frolov, and I. Titov. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. *arXiv preprint arXiv:1701.02593*, 2017.

M. A. Nomani and D. M. Sharma. Towards building semantic role labeler for indian languages. *FC Kochi center on Intelligent Systems (KCIS), IIIT-Hrdrabad, India*, 2016.

L. A. Pizzato and D. Mollá. Indexing on semantic roles for question answering. In *Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering*, pages 74–81. Association for Computational Linguistics, 2008.

V. Punyakanok, D. Roth, W.-t. Yih, and D. Zimak. Semantic role labeling via integer linear programming inference. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1346. Association for Computational Linguistics, 2004.

V. Punyakanok, D. Roth, and W.-t. Yih. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2): 257–287, 2008.

M. Roth and M. Lapata. Neural semantic role labeling with dependency path embeddings. *arXiv preprint arXiv:1605.07515*, 2016.

A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

D. Shen and M. Lapata. Using semantic roles to improve question answering. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007.

M. Surdeanu and J. Turmo. Semantic role labeling using complete syntactic analysis. In *Proceedings of the Ninth Conference on Computational Natural*

Language Learning, pages 221–224. Association for Computational Linguistics, 2005.

O. Täckström, K. Ganchev, and D. Das. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41, 2015.

A. Vaidya, J. D. Choi, M. Palmer, and B. Narasimhan. Analysis of the hindi proposition bank using dependency structure. In *Proceedings of the 5th Linguistic Annotation Workshop*, pages 21–29. Association for Computational Linguistics, 2011.

N. Xue and M. Palmer. Calibrating features for semantic role labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 2004.

M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

J. Zhou and W. Xu. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1127–1137, 2015.

Knowledge Building via Optimally Clustered Word Embedding with Hierarchical Clustering

Nadeesha Pathirana
Sandaru Seneviratne
Rangika Samarawickrama
Shane Wolff
Charith Chitraranjan
Uthayasanker Thayasivam
Department of Computer Science and Engineering,
University of Moratuwa,
Sri Lanka.
nadeesha.14@cse.mrt.ac.lk
sandaru.14@cse.mrt.ac.lk
rangika.14@cse.mrt.ac.lk
shanewolff.14@cse.mrt.ac.lk
charithc@cse.mrt.ac.lk
rtuthaya@cse.mrt.ac.lk

Tharindu Ranasinghe
CodeGen International (Pvt.) Ltd.
tharindu.10@cse.mrt.ac.lk

Abstract

Given a particular domain, drawing out information from a vast amount of data is not an easy task. Data may be adequate and abundant but analysis of data requires a great deal of work. Therefore, manual construction of a knowledge base on any particular domain is more time consuming and it requires much human intervention. The process can be semi-automated by effectively using word embedding to identify the semantics. The words can then be clustered using hierarchical clustering. This study proposes a novel approach on how knowledge discovery can be semi-automated for the restaurant domain by effectively identifying the optimal number of clusters from hierarchical clustering of words extracted from restaurant reviews.

1 Introduction

To understand a particular domain, identifying domain specific concepts is a must and a comprehensive knowledge base paves the way to do that. As we are considering restaurant domain in our research, we should identify concepts like food, staff, atmosphere, sub categories of concepts and the relationships among them. Since, identifying these things manually and creating a knowledge base based on them is an exhaustive process, automating it will be very convenient. This requires a model which can capture meanings, similarities and relationships of given data. As word vector embedding models are capable of capturing semantic relationships, the study proposes a novel method to use them to identify concepts via hierarchical clustering by processing user reviews which paves the way to build the knowledge base for the restaurant domain.

Automating the construction of knowledge base includes developing a computational model which could capture not only the meaning but also the relative meaning and similarity among the words in a corpus which is also considered as an important task in the field on Natural Language Processing. This requirement can be modeled as a vector space which represents words as vectors with high dimensions. These vectors are clustered using hierarchical clustering to obtain a hierarchy of the concepts in the domain. In this research, we have proposed a novel way of obtaining the optimal number of clusters from the hierarchical clustering using the silhouette score. Through the obtained clusters, an ontology for the restaurant domain can be built which can be used in constructing a comprehensive knowledge base.

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 69–73

2 Background and Related Work

2.1 Word Vector Embedding

Word vector embedding is the NLP and feature learning technique where the words from a vocabulary are mapped to vectors of real numbers. Word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA) are popular word vector models. Since LSA and LDA perform analogy tasks poorly, Word2Vec and GloVe models were used in this research.

Word2Vec which is considered to be good in analogy tasks (Pennington et al., 2014) is an Artificial Neural Network (ANN) based solution which takes the surrounding words and target words into consideration. There are two main architectures in Word2Vec as Skip Gram (Melamud et al., 2015) and Continuous Bag of Words (CBOW) (Goldberg and Levy, 2014). In Skip Gram model the target word is fed to ANN as the input to predict the context words whereas in CBOW, a word is predicted using the context words in surrounding. GloVe is a model which uses matrix factorization method to exploit statistical information which is similar to LSA and LDA, but it combines the benefits of the Word2Vec Skip Gram model to result a more accurate prediction (Pennington et al., 2014) unlike LSA and LDA.

2.2 Clustering

Clustering is the task of grouping objects based on their characteristics such that they have high intra-class similarity and low inter-class similarity (Srivastava et al., 2000). Among all the clustering methods, the consideration was focused on hierarchical clustering. Among two types of hierarchical clustering approaches which are divisive (Savaresi et al., 2002) and agglomerative (Beeferman and Berger, 2000) the latter was preferred.

Obtaining the optimal number of clusters from a dendrogram while preserving the cluster quality is challenging. Knowing the optimal clusters is vital for the study as they represent the concepts in the domain. Researchers have found many indices which measure the quality of the clusters. In this approach, the average silhouette index (Liu et al., 2010) was used to measure the cluster quality from which the optimal number of clusters were obtained. The optimality of the clusters not only depends on the number of overall clusters obtained, but also the composition of the clusters plays a role in aspect of quality. Fine tuning the word embedding models assists the process of capturing the semantics best and therefore improving the model may result in improved cluster composition.

3 Methodology

This section provides a complete understanding of the procedure that has been followed to conduct the research. Each subsection below is a component of the overall methodology.

3.1 Data Collection

Since the initial goal of the study was to build a restaurant domain specific, comprehensive knowledge base, the domain related data have been collected using nearly a half a million of user reviews. Since user reviews cover aspects like food, staff and environment with respect to a restaurant those reviews can be used as a reliable source.

3.2 Data Preprocessing

Three collections of data; user reviews, the Stanford–10–million–word set and the w2v_gbg dataset were used. Reviews were preprocessed before generating the word embedding models. All non–alphabetical characters and stop words were removed (stop words carry insignificant meaning compared to keywords in a domain) (Srividhya and Anitha, 2010). All the letters were converted to the lower case so as to prevent confusion when recognizing similar words.

3.3 Frequent Noun Extraction

Domain related concepts can be identified by analyzing the most frequently occurring nouns in the review text corpus. Therefore, to extract nouns in the text, parts of speech (POS) tagging was performed. To enhance the accuracy of noun extraction process, two different libraries were used for POS tagging and a candidate word was labelled as a noun only if both libraries recognized it as a noun. Then, most frequently occurring ten thousand nouns were selected by sorting the frequency distribution of noun occurrence in the text corpus in descending order and selecting the first ten thousand nouns. Further, cosine distance between word vectors of a given noun and a domain related word was used to filter out closely related domain specific nouns. A threshold was established from which a noun was recognized as domain related if the aforementioned distance was below the thresh-

Data set	Model No.	Window Size	Context
Stanford	G1	15	Symmetric
Stanford	G2	20	Asymmetric
w2v_gbg	G3	15	Symmetric
w2v_gbg	G4	20	Asymmetric

Table 1: Variable Parameters of GloVe Models

Data set	Model No.	Architecture
w2v_gbg	W1	Skip Gram
w2v_gbg	W2	CBOW
Stanford	W3	Skip Gram
Stanford	W4	CBOW

Table 2: Variable Parameters of Word2Vec Models

old. These nouns were used later on for hierarchical clustering (Beil et al., 2002).

3.4 Word Embedding – Word2Vec and GloVe

Words that are related have similar vector representations. Therefore, it is easy to identify relationships among words using word vectors. To obtain word vectors, word embedding models which were built using Word2Vec and GloVe algorithms were used. When generating a word embedding model, vector representation of a word differs from model to model depending on the parameter assignment. Therefore, few different models were built and hierarchical clustering was performed. Based on the quality of the dendrogram (discuss later in topic 3.5), the most accurate model was selected. Table 1 contains variable parameters of Glove models. The parameters vector size = 300, minimum word count = 5, and number of iterations = 15 were kept fixed throughout different GloVe models. Table 2 contains variable parameters of Word2Vec models. The parameters vector size = 300, window size = 5, and minimum word count = 5 were kept fixed throughout different Word2Vec models.

3.5 Hierarchical Clustering

Performing clustering on a data set will result clusters with similar objects. Therefore, it is easy to identify the concept that each cluster represents. Agglomerative hierarchical clustering was performed on domain specific frequent noun set (Beil et al., 2002). Since the exact number of concepts required to explain the restaurant domain is unknown, hierarchical clustering was used as it does not require the number of clusters beforehand. The rationale for choosing the agglomerative approach was that it is less time consuming compared to divisive.

When building the hierarchy, different linkage methods and metrics can be used. Based on them, for the same word embedding model, different visual representations (dendrograms) can be obtained. Cophenetic coefficient (C) was used to evaluate the dendrogram. It indicates how much the dendrogram preserves the word similarity compared to the actual word similarity. Accuracy of the dendrogram depends on the accuracy of the model. Therefore, C can be used to measure the accuracy of the model too. As C increases and approaches +1, the quality of the dendrogram improves. The results are shown in the results section.

3.6 Identifying Clusters

The most crucial step in the methodology is to identify the optimal number of clusters. Even though, flat clusters could be obtained by setting a horizontal line at a required distance in the dendrogram, the optimality cannot be guaranteed. Therefore, the overall cluster quality should be measured while obtaining the clusters. Silhouette index measures the quality of a cluster compared to other clusters. Calculating scores for each and every cluster and then finally averaging them results in a measure which evaluates the overall cluster setting. The method can be enumerated on all possible numbers of clusters and the number which corresponds to the highest average score is treated as optimal. In equation (1), $\text{avgSI}(k)$ denotes the average silhouette index for k number of clusters and $\text{SI}(i)$ denotes the silhouette index for the i^{th} cluster and in equation (2), k_{opt} denotes the number of optimal clusters and n is the total number of words considered (maximum possible clusters).

$$\text{avgSI}(k) = \frac{1}{k}\sum_{i=1}^{k}\text{SI}(i) \tag{1}$$

$$k_{\text{opt}} = \underset{2\leq k<n}{\arg\max}[\text{avgSI}(k)] \tag{2}$$

4 Results

When clustering, models built with Stanford data set produced some clusters with words that are unrelated with reference to the restaurant domain. E.

Model No.	Average	Complete	Cosine
W1	0.7336	0.4996	0.5488
W2	0.4837	0.3919	0.3919
W3	0.4942	0.2493	0.4537
W4	0.3985	0.2319	0.0604
G1	0.2782	0.2782	0.0493
G2	0.3683	0.0239	0.0562
G3	0.3786	0.1936	0.1005
G4	0.4235	0.1326	0.1243

Table 3: Cophenetic scores for different linkage methods

g. the word *apple* with words *blackberry*, *mobile*, *wifi*, *gsm* etc. The reason is that word *apple* was frequently used to represent the brand name **Apple** in the data set. Since, this research is relevant to the restaurant domain, we need to identify *apple* as a fruit. Therefore, w2v_gbg data set was attempted which gave better results.

When the distance of two word vectors is obtained via euclidean distance metric, the result includes a portion of the vector magnitude and the direction both. The vector magnitude is affected by the frequency of occurring the word in the text corpus. The vector direction attempts to capture the semantic meaning of the word. Since we are interested at the semantic similarity/dissimilarity, euclidean distance metric may not fully capture the requirement. However, the cosine distance captures the directionality of the vectors and therefore the distance result incorporates much more semantic information of the words compared to euclidean distance. According to Saraçli et al. (2013), average linkage is the best method to generate the dendrogram. Table 3 summarizes the cophenetic scores obtained for different models under different linkage methods of dendrogram.

Based on the results in table 3, the most accurate model for the hierarchical clustering is W1 with the highest cophenetic coefficient of 0.7336 with the parameters; algorithm = Skip Gram, vector size = 300, window size = 5, and minimum word count = 5. Figure 1 and 2 depict dendrogram extracts of coffee types and alcoholic beverages respectively.

The average silhouette index was calculated for all possible numbers of clusters. The maximum average index was reached with 1779 clusters with the magnitude of 0.177. Therefore, the optimal number of clusters would be a number around 1780.

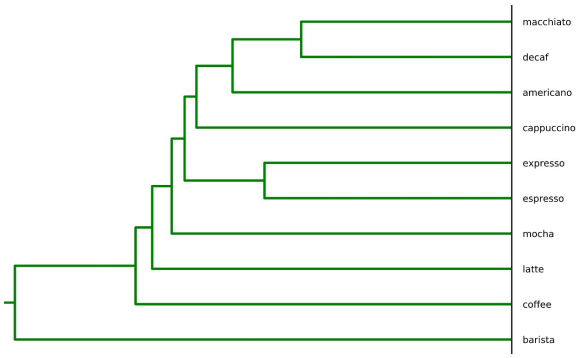


Figure 1: Dendrogram extract of coffee types

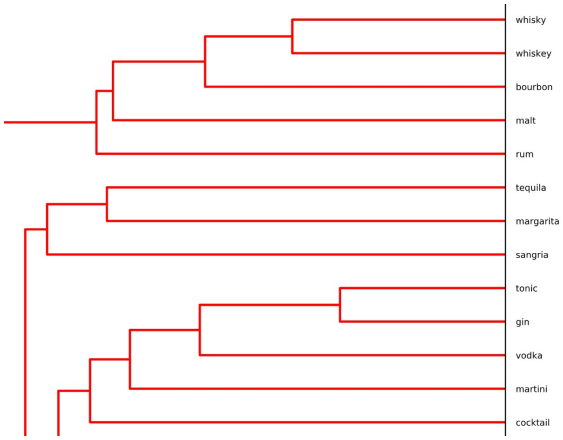


Figure 2: Dendrogram extract of alcoholic beverages

5 Conclusion and Future Work

This study has demonstrated the process of obtaining the optimal number of clusters through the construction of hierarchical clustering of concepts for the restaurant domain and it is assisted by word embedding technique. Hence, the process is automated to a great extent while completely avoiding the hassle caused by manually creating the knowledge base. The proposed method is general and flexible, and can be adapted to any other domain as well. The research focuses on how more accurate clusters/concepts can be obtained through hierarchical clustering. The average silhouette score has been used to obtain the optimal number of clusters. The procedure can be extended to obtain a comprehensive knowledge base for the restaurant domain by building an ontology from the identified clusters.

References

Doug Beeferman and Adam Berger. 2000. Agglomerative clustering of a search engine query log. In *ACM SIGKDD international conference on Knowledge discovery and data mining*. https://dl.acm.org/citation.cfm?id=347176.

Florian Beil, Martin Ester, and Xiaowei Xu. 2002. Frequent term-based text clustering. In *ACM SIGKDD international conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, pages 436–442. https://dl.acm.org/citation.cfm?id=846188.

Yoav Goldberg and Omer Levy. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *ArXiv e-prints* https://arxiv.org/pdf/1402.3722.pdf.

Yanchi Liu, Zhongmou Li, Hui Xiong, Xuedong Gao, and Junjie Wu. 2010. Understanding of internal clustering validation measures. In *IEEE International Conference on Data Mining*. IEEE. https://ieeexplore.ieee.org/document/5694060/.

Oren Melamud, Omer Levy, and Ido Dagan. 2015. A simple word embedding model for lexical substitution. In *NAACL-HLT*. Association for Computational Linguistics. http://www.aclweb.org/anthology/W15-1501.

Tomas Mikolov, Kai Chen, Greg. Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *ArXiv e-prints* https://arxiv.org/pdf/1301.3781.pdf.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics. https://www.aclweb.org/anthology/D14-1162.

Sinan Saraçli, Nurhan Doğan, and İsmet Doğan. 2013. Comparison of hierarchical cluster analysis methods by cophenetic correlation. *Journal of Inequalities and Applications* https://link.springer.com/content/pdf/10.1186%2F1029-242X-2013-203.pdf.

Sergio Savaresi, Daniel Boley, Sergio Bittanti, and Giovanna Gazzaniga. 2002. Cluster selection in divisive clustering algorithms. In *SIAM International Conference on Data Mining*. https://epubs.siam.org/doi/pdf/10.1137/1.9781611972726.18.

Jaideep Srivastava, Robert Cooley, and Mukund Deshpande. 2000. Web usage mining: Discovery and applications of usage patterns from web data. In *NAACL-HLT*. Association for Computational Linguistics, pages 12–23. https://dl.acm.org/citation.cfm?id=846188.

V Srividhya and R Anitha. 2010. Evaluating preprocessing techniques in text categorization. *International Journal of Computer Science and Application* http://sinhgad.edu/ijcsa-2012/pdfpapers/1_11.pdf.

Machine Learning Approaches for Amharic Parts-of-speech Tagging

Ibrahim Gashaw
Mangalore University
Mangalagangotri, Mangalore-574199
ibrahimug1@gmail.com

H L Shashirekha
Mangalore University
Mangalagangotri, Mangalore-574199
hlsrekha@gmail.com

Abstract

Part-of-speech (POS) tagging is considered as one of the basic but necessary tools which are required for many Natural Language Processing (NLP) applications such as word sense disambiguation, information retrieval, information processing, parsing, question answering, and machine translation. Performance of the current POS taggers in Amharic is not as good as that of the contemporary POS taggers available for English and other European languages. The aim of this work is to improve POS tagging performance for the Amharic language, which was never above 91%. Usage of morphological knowledge, an extension of the existing annotated data, feature extraction, parameter tuning by applying grid search and the tagging algorithms have been examined and obtained significant performance difference from the previous works. We have used three different datasets for POS experiments.

1 Introduction

POS tagging is the process of assigning the part of speech categories to each and every word in a sentence. In many NLP applications such as word sense disambiguation, information retrieval, information processing, parsing, question answering, and machine translation, it is considered as one of the basic but necessary tool that could be utilized in computational linguistics analysis and automation applications (Antony and Soman, 2011).

Existing POS tagger approaches can be classified into: linguistic (rule-based), statistical/machine-learning and hybrid approaches.

Linguistic approachs: Most POS taggers arrange linguistic knowledge systematically as a set of rules (or constraints) written by linguists that range from a few hundred to several thousand, and usually require a high cost for experts and consume time (Màrquez et al., 2000).

Statistical/Machine Learning approaches: These approaches use frequency or probability to tag words in a text. With the simplest Statistical tagger, the ambiguity of words established on the probability that the word occurs alongside a particular tag can be resolved. Statistical approach involves some kind of learning (supervised or unsupervised) parameters of the model from a training corpus (Radziszewski, 2013).

Hybrid approaches: It includes transformation-based approach that combines rule-based approach and statistical approach. These approaches helps to achieve a significant improvement of POS performance, since it can combine necessary features from statistical and linguistic based approaches (El Hadj et al., 2009).

Every human language poses its own challenges and requires specific methods. Amharic is also one of the families of morphologically rich languages that has major challenges related to POS tagging task.

All proposed POS taggers were based on ELRC Tagset, developed by different individuals. This paper addresses the various developments in POS-taggers and POS-tagset for the Amharic language, which is very essential computational linguistic tool needed for many NLP applications. We focused on extending existing annotated data (ELRC tag-set), constructing new tag-set and then implementing machine learning methods that have been recently applied to solve POS problems of Amharic Language.

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 74–79

2 Previous work

Several attempts have been made in the past to develop POS algorithms for Amharic Language. Some of these works are as follows.

Getachew (2001), attempted to develop a Hidden Markov Model (HMM) based POS tagger using 23 POS tags from 300 words. Since the tag-set and data are very small, the tagger does not have the capability of predicting the POS tag of unknown words.

Adafre (2005), developed a POS tagger using Conditional Random Fields (CRF) and abstract tag-set consisting of 10 tags and obtained overall accuracy of 74% on a manually annotated text corpus of five Amharic news articles (1000 words). The small amount of annotated data leads to drastic impact on tagging accuracy.

Gamback et al. (2009), compared three tagging strategies; HMM, Support Vector Machines (SVM) and Maximum Entropy (ME) using the manually annotated corpus developed by Demeke and Getachew (2006) at the Ethiopian Language Research Center (ELRC) of Addis Ababa University. Since the corpus contained few errors and tagging inconsistencies, they have cleaned the corpus. They obtained the average accuracies (after 10-fold cross validation) of 85.56%, 88.30%, and 87.87% for the HMM, SVM, and ME-based taggers respectively for the ELRC tag-set.

Tachbelie and Menzel (2009), conducted POS tagging experiments for Amharic using uncleaned ELRC corpus in order to use POS information in language modeling. They developed Trigrams'n'Tags (TnT) and SVM-based taggers and compared in terms of performance, tagging speed as well as memory requirements. The results of their experiments show that with respect to accuracy, SVM-based taggers perform better than TnT-based taggers although TnT-based taggers are more efficient with regard to speed and memory requirements. This work lacked a reference allowing for an evaluation of the quality of the annotations that may highly affect the performance of taggers.

Gebre (2010), attempts to improve the performance of Amharic POS tagger based on CRF, SVM, Brill and HMM. Cleaning up ELRC tag-set to minimize the pre-existing tagging errors and inconsistencies can increase the performance of the POS tagger. With 10-fold cross validation they have obtained average accuracy of 90.95%, 90.43%, 87.41%, and 87.09% for CRF, SVM, Brill and TnT taggers respectively. Even though they obtained good accuracy the precision and recall reported in this paper is very far from the average accuracy. For example, by using CRF tagger they have obtained 60% recall and 67% precision and using SVM 64% recall and 68% precision.

3 Amharic Language

Amharic is the second most widely spoken Semitic language in the world, after Arabic. It is characterized by complex, productive morphology, with a basic word-formation mechanism, root and pattern (Shashirekha and Gashaw, 2016).

The typical clause order in Amharic is noun + object + verb. Nouns may denote gender, number, definiteness, case, and direct object status by affixes prefixes and suffixes, predominately suffixes. Amharic nouns may have a masculine or feminine gender. Suffixes are added to denote a masculine or feminine noun gender. Some nouns may have both masculine and feminine gender, while other nouns may only have one gender. The feminine gender is used to indicate female as well as the smallness (Degsew, 2014).

4 Proposed Approach

Sentence and word tokenization is performed for unannotated Quran and Bible texts before part of tagging process. Since Amharic annotated corpus (ELRC) is limited with only news domain and cleaned version of this dataset is not available, ELRC data is cleaned. The tag-set in ELRC is only 31 tags. It cannot give much information to reliably develop NLP applications. Therefore, ELRC tag-set is extended from 31 to 51 and then new corpus is constructed from Quran and Bible texts including the extended ELRC annotated data. For training and testing algorithms, data should be splited into training and test set, then morphological features are extracted from the training set for CRFSuit tagger only. After-all training and testing are conducted on the three tagging algorithms for each tag-set.

Particularly we applied machine learning approaches for the POS tagging task, and it can be easily interpreted as a classification problem. In this POS task, the limited set of tags are identified from three Corpora and the training examples

are the occurrences of the words along with the respective POS category in the context of appearance. A general representation of the POS tagging process is Shown in Figure 1.

We adopt CRF model, which has widely been

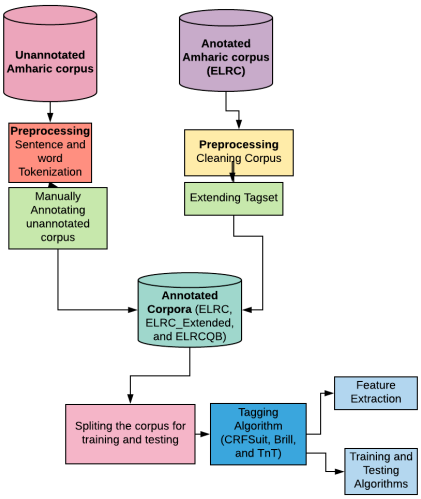


Figure 1: General framework of the proposed approach

used in several basic NLP tasks. It is a conditional model that models the conditional probability distribution of tags ($t_1...t_k$) given observation sequences of words ($w_1...w_k$) in the sentences i.e. $P(t...t_k|w_1...w_k)$. The probability of transition between tags is depends on the previous and next observations. This enables reasoning based on wide contexts, which seems especially important in the case of POS tagging tasks. For large and structured tag-sets, CRF work well with many features that may be mutually dependent (Lafferty et al., 2001).

Linear−chain CRF is the most popular class of CRFsuit suitable for tagging. CRFSuit training consists of estimation of weight values. A high weight value indicates that strong evidence has been found to support the relation between observations and tags as expressed by the feature (Radziszewski, 2013). Tagging with a trained CRFsuit consists in finding a tag sequence that maximizes the conditional probability. The optimization algorithms used in these work is the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm (Saputro and Widyaningsih, 2017) which is employed for solving high-dimensional minimization problems in scenarios where both the objective function and its gradient can be computed analytically. L-BFGS algorithm stores information about the spatial displacement and the change in gradient and uses them to estimate a search direction without storing or computing the Hessian explicitly (Coppola and Stewart, 2014).

4.1 Feature extraction

Since Amharic is morphologically rich language, there are a lot of morphological features which enables the POS tagger to predict correctly. In a CRFSuit, each feature function is a function that takes in as input: a sentence s, the position i of a word in the sentence, either the word comes first/last, has hyphen, is current word/previous word/next word digit, alphanumeric, prefix-1, prefix-2, prefix-3, suffix-1, suffix-2, suffix-3, Previous-1 word tag, and Previous-2 word tag. For each such value configuration, a separate function must be provided in advance. In order to train and test the POS tagger, we define a function that can extract all the above features and then used to input in CRFSuit feature function.

4.2 Dataset description

The dataset used in this study are categorized in to three, ELRC annotated corpus that contains 210,000 words (Demeke and Getachew, 2006), extended re-tagged corpus of ELRC, and the newly annotated corpus of the Amharic translation of Quran, and Bible.

In the first domain, the tag-set is based on 11 basic tags, most of which have further been refined to provide more linguistic information, thus increasing the tag-set to 31.

Even though, (Gebre, 2010) cleaned ELRC tagged corpus, we couldn't get the cleaned one. Therefore, we have enforced to clean again by following the strategies used to clean in this work.

Since Amharic is morphologically complex language, 31 tags of ELRC tag-set cannot give much information to reliably develop NLP applications. Some tags that may be critical depending on the target application are missing. In detail, the limitation of ELRC tag-set is reported by (Gebre, 2010). Furthermore, we extended ELRC tag-set from 31 tags to 51 tags by adding S for those tags with plural numbers such as Noun with plural numbers (NS) and the addition of preposition and conjunction with adverbs then we called this dataset ELRC-Extended. The third category is the extended ELRC tagset plus manually tagged Quran

and Bible Documents by taking ELRC as a base, we call this new dataset ELRCQB which contains 62 tags. ELRCQB dataset size is 33,940 sentences (440,941 words). For Example the distribution of ELRC-Extended tag-set is shown in figure 2.

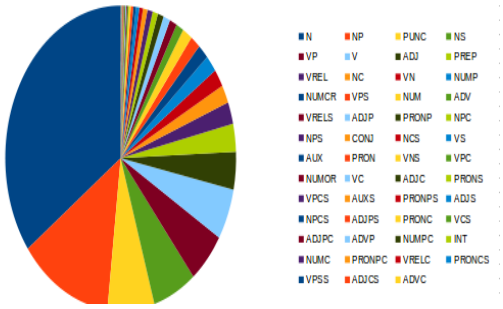


Figure 2: ELRCExtended tag-set distribution

5 Experiments and Results

This section presents the experiments validating the three machine learning algorithms (Brill, TnT, and CRFSuit Taggers) implemented for Amharic POS Tagging task. We use sklearn-crfsuite which is a CRFsuite (python-crfsuite) wrapper that provides scikit-learn compatible sklearn-crfsuite.CRF estimator to train and test our POS tagger. 10-Fold cross-validation is applied for training and then evaluating all tagging techniques for all three corpora. 10-fold cross-validation data for all tagsets is shown on table 2 with the information of known and unknown words of the testing data for each fold.

The results obtained by applying the three different tagging strategies are shown in Table 2. TnT tagger and Brill tagger performs almost the same. CRFSuit Tagger achieves the best scores of all three taggers. Because the features extracted from the tag-sets enables the system to predict the words tag even if it is not in training data. To handle unknown words for Brill and TnT tagger, we used n-gram tagger as back-of tagging strategies that assign a maximum tag appeared in the test set.

All approaches are evaluated using confusion matrix. In this POS tagging problem, the confusion matrix contains 62 rows and 62 columns, 52 rows and 52 columns, 31 rows and 31 columns for ELRCQB, ELRC-Extended and ELRC tag-sets of all tagger. But due to a large number of tags that may not have good visibility, we showed only the first top 20 tags of confusion matrix for the best score. Confusion matrix helps to indicate correctly classified and wrongly classified elements of each class. For example, in figure 5 the proposed approach (CRFSuit Tagger), the tag N, 9388 of 9930 are classified correctly but the remaining 542 are misclassified as different tags. The vertical lines associated with each confusion matrix indicates the elements in the class with a maximum number of elements predicted.

The base for our work is (Gebre, 2010), that yields good performance for different machine learning approaches using 10 fold cross-validation technique, which is the overall accuracy of 90.95, 90.43, 87.41, 87.09 for CRF, SVM, Brill, and TnT taggers respectively on cleaned ELRC tagset only. Even though overall accuracy is reported as above the precision and recall result reported in his work is less compared to our precision and recall results showed in table 3.

The main contribution of this work is extending ELRC tagset, constructing new tagset from Quran and Bible document and parameter tuning by selecting the best parameter of Sklearn-CRFSuit through grid searching which is C1:0.064 and C2:0.002. We achieved the overall average accuracy of 86.44, 95.87, and 92.27 for ELRC, ELEC-Extended and ELRCQB tagsets respectively. As the result indicated extending the tag-set increased the performance by 9.43 which is very significant. While the domain of the tag-set increasing the number of unknown words also increased and the style of writing in Quran and Bible is different, then it creates misclassification. Even though increasing the tag-set size has its own advantage in machine learning approaches in general, in this work the noise from domain difference creates 3.6% performance difference between ELRC-Extended and ELRQB tag-sets.

6 Conclusion

In this paper, we have described three machine learning approaches for automatic tagging of Amharic text processing. The models described here are very simple and efficient for automatic tagging. The extended and newly constructed tag-sets have contributed to the high performance of our proposed approach and it will contribute to the reliable development of applications of machine translation, information retrieval, information extraction and speech synthesis/recognition. The

Table 1: 10-Fold Cross Validation Data

Fold	# ELRC Words				# ELRC_Extended Words				# ELRCQB Words			
	Training	Testing			Training	Testing			Training	Testing		
		Known	Unknown	Total		Known	Unknown	Total		Known	Unknown	Total
1	180788	18102	2744	20846	185091	18433	2646	21079	411643	27147	4245	31392
2	181435	17637	2562	20199	185447	18233	2490	20723	413002	26065	3968	30033
3	182222	17155	2257	19412	186381	17634	2155	19789	414879	24716	3440	28156
4	181722	17532	2380	19912	185532	18344	2294	20638	416764	22819	3452	26271
5	180956	18188	2490	20678	185089	18659	2422	21081	376787	57808	8440	66248
6	182679	16867	2088	18955	186892	17233	2045	19278	370077	64255	8703	72958
7	181368	17841	2425	20266	185415	18409	2346	20755	370601	63842	8592	72434
8	181189	17965	2480	20445	185100	18692	2378	21070	407013	30114	5908	36022
9	181735	17588	2311	19899	185747	18208	2215	20423	405709	31495	5831	37326
10	180612	18639	2383	21022	184836	19000	2334	21334	400840	33654	8541	42195
Average	181470.6	17751.4	2412	20163.4	185553	18284.5	2332.5	20617	398731.5	38191.5	6112	44303.5

Table 2: Average 10-Fold Accuracy of Brill, TnT and CRFSuit Taggers

Tagger	ELRC			ELRC-Extended			ELRCQB		
	Known words	Unknown words	Overall	Known words	Unknown words	Overall	Known words	Unknown words	Overall
Brill	89.185	25.97	81.627	99.997	46.206	93.876	97.359	33.409	88.539
TnT	89.889	25.969	82.25	99.997	46.214	93.877	97.491	33.409	88.661
CRFSuit	87.883	75.79	86.442	99.069	70.872	95.868	96.408	65.634	92.242

Table 3: CRFSuit Tagger Best score of Average precision, recall and f-1score on ELRC, ELRC-Extended, and ELRCQB tag-set

Tag-set	**precision**	**recall**	**f1-score**	**support**
ELRC	0.902	0.898	0.899	20445
ELRC-Extended	0.97	0.97	0.97	21070
ELRCQB	0.951	0.951	0.951	36022

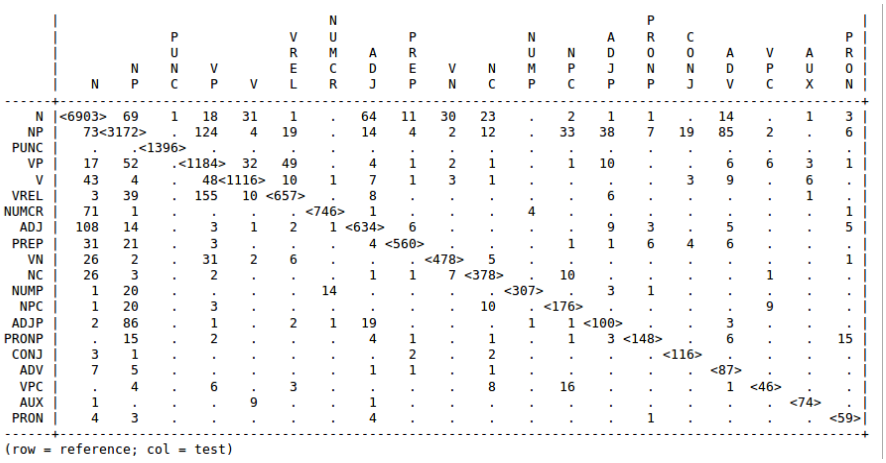


Figure 3: Confusion matrix of Top 20 Tags best score for ELRC tagset

best performances are achieved for the CRFSuit learning model along with the important morphological features extracted from the training set.

All the tag-sets we have used in this work lacks expert knowledge. Therefore, it should be standardized to obtain enhanced performance. It is very limited to identify names of people and places, which is critical for information extraction. The presence of a proper noun tag is even more important in the context of Amharic, but the idea of letter case distinction does not exist and most Ethiopian names are just normal words in the language. Thus, most proper nouns that are easily recognized in English by the case of the initial letter cannot be recognized in Amharic.

References

Sisay Fissaha Adafre. 2005. Part of speech tagging for amharic using conditional random fields. In *Proceedings of the ACL workshop on computational approaches to semitic languages*, pages 47–54. Association for Computational Linguistics.

PJ Antony and KP Soman. 2011. Parts of speech tagging for indian languages: a literature survey. *International Journal of Computer Applications (0975-8887)*, 34(8):22–29.

Antonio Coppola and Brandon M Stewart. 2014. lbfgs: Efficient l-bfgs and owl-qn optimization in r.

Tihitina Petros Degsew. 2014. *Modeling and Designing Amharic Query System To Bilingual (English-Amharic) Databases*. Ph.D. thesis, Thesis submitted to the school of graduate studies of the Addis Ababa University.

Girma A Demeke and Mesfin Getachew. 2006. Manual annotation of amharic news items with part-of-speech tags and its challenges. *Ethiopian Languages Research Center Working Papers*, 2:1–16.

Yahya El Hadj, I Al-Sughayeir, and A Al-Ansari. 2009. Arabic part-of-speech tagging using the sentence structure. In *Proceedings of the Second International Conference on Arabic Language Resources and Tools, Cairo, Egypt*.

Björn Gambäck, Fredrik Olsson, Atelach Alemu Argaw, and Lars Asker. 2009. Methods for amharic part-of-speech tagging. In *Proceedings of the First Workshop on Language Technologies for African Languages*, pages 104–111. Association for Computational Linguistics.

Binyam Gebrekidan Gebre. 2010. *Part of speech tagging for Amharic*. Ph.D. thesis, University of Wolverhampton Wolverhampton.

Mesfin Getachew. 2001. *Automatic Part of Speech Tagging for Amharic Language: An Experiment Using Stochastic Hidden Markov (HMM) Approach*. Ph.D. thesis, Master Thesis at School of Information Studies for Africa, Addis Ababa.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Lluís Màrquez, Lluis Padro, and Horacio Rodriguez. 2000. A machine learning approach to pos tagging. *Machine Learning*, 39(1):59–91.

Adam Radziszewski. 2013. A tiered crf tagger for polish. In *Intelligent tools for building a scientific information platform*, pages 215–230. Springer.

Dewi Retno Sari Saputro and Purnami Widyaningsih. 2017. Limited memory broyden-fletcher-goldfarb-shanno (l-bfgs) method for the parameter estimation on geographically weighted ordinal logistic regression model (gwolr). In *AIP Conference Proceedings*, volume 1868, page 040009. AIP Publishing.

HL Shashirekha and Ibrahim Gashaw. 2016. Dictionary based amharic-arabic cross language information retrieval. *Computer Science & Information Technology*, page 49.

Martha Yifiru Tachbelie, Solomon Teferra Abate, and Laurent Besacier. 2011. Part-of-speech tagging for underresourced and morphologically rich languagesthe case of amharic. *HLTD (2011)*, pages 50–55.

Martha Yifiru Tachbelie and Wolfgang Menzel. 2009. Amharic part-of-speech tagger for factored language modeling. In *Proceedings of the International Conference RANLP-2009*, pages 428–433.

English to Bodo Statistical Machine Translation System Using Multi-domain Parallel Corpora

Saiful Islam
Dept. of Computer Science
Assam University
Silchar, India

Ismail Hussain
Dept. of Bodo
Gauhati University
Guwahati, India

Bipul Syam Purkayastha
Dept. of Computer Science
Assam University
Silchar, India

Abstract

Parallel corpus is a primary resource for most of the applications of Natural Language Processing (NLP) like Machine Translation (MT) and CLIR. Statistical Machine Translation (SMT) is a highly successful and popular approach of MT that can produce high-quality translation results using a huge amount of bilingual parallel corpus. This paper primarily focuses on the development of English to Bodo SMT system using multi-domain English-Bodo Parallel Text Corpus (E-BPTC). The SMT system has been developed using the Phrase-based SMT approach for the different domains of E-BPTC, namely Tourism, News, Health, General and Agriculture. The translation accuracy of the different domains of E-BPTC in the SMT has been evaluated using the Manual and Automatic evaluation techniques.

1 Introduction

Machine translation is the most important application of NLP that translates texts from one natural language to another automatically and quickly. Nowadays, MT is a very challenging research task globally in the field of NLP. It is a very difficult task due to some challenges in natural languages like word order and word ambiguity. The approaches of MT can be classified into different categories (Antony, 2013; Islam & Purkayastha, 2018) as shown in Figure 1.

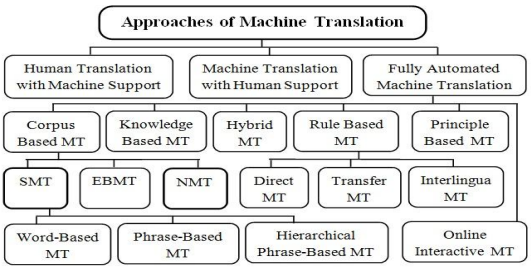


Figure 1: Approaches of MT

At present, the most popular and state-of-the-art approaches of MT are SMT and NMT (Neural Machine Translation). The SMT has gained tremendous potential globally in the research community as well as in the commercial sectors. In 1949, Warren Weaver introduced the first concepts of SMT approach (Kathiravan et al., 2016). The SMT is the best technique of MT for reducing word ambiguity problems in the natural languages. It requires less linguistic knowledge and can reduce human efforts (Koehn, 2009). It is classified as Word-based SMT, Phrase-based SMT and Hierarchical Phrase-based SMT.

Bodo is one of the major spoken languages of North-East India (Islam et al., 2017). It is a recognized language of India and is an official language of Bodoland Territorial Council (Assam). The Bodo language is written using Devanagari script. It is a low resource language and the word order of it is Subject+ Object + Verb.

English is an International human language and is primarily spoken by the people of many countries, such as Australia, United Kingdom and the United States. English is an associate official language of India (Islam and Purkayastha, 2018). The English language is written using Latin script. It is a high resource language and the word order of it is Subject + Verb + Object.

2 Related Work

A large number of SMT systems and parallel corpora have been developed and constructed for popular natural languages as well as for Indian languages. Some of them are discussed below.

2.1 Parallel Corpus

Lots of parallel corpus has been built globally for popular natural languages. Some of the parallel corpora are briefly discussed as follows:

Bible corpus: The corpus was constructed from the translation of the Bible. It is a multilingual

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 80–86

parallel text corpus and contains 100 natural languages. The corpus is freely available online[1].

English-Kazakh parallel corpus: The corpus was constructed at Al-Farabi Kazakh National University, Kazakhstan by (Kuandykova et al., 2014) for the English⇔Kazakh SMT system.

Europarl corpus: The corpus was constructed at the University of Edinburgh, Scotland, UK by (Koehn, 2005) for the SMT system. The corpus is freely available online[2].

OPUS Corpus: The OPUS is a multilingual parallel corpus that contains 60 different languages. The corpus is freely available online[3].

UM corpus: The corpus was constructed at the University of Macau, China by (Tian et at., 2014) for the SMT system. It is a multi-domain English-Chinese parallel text corpus.

Some of the parallel corpora which have been constructed for English and Indian natural languages are discussed as follows:

TDIL corpus: The TDIL[4] (Technology Development for Indian Languages) programme has constructed different domains of parallel corpora. Some of the parallel corpora exist in this corpus are English-Assamese, English-Bodo, English-Hindi, Hindi-Punjabi and Hindi-Urdu.

EMILLE/CIIL Corpus: The EMILLE (Enabling Minority Language Engineering)/CILL (Central Institute of Indian Languages) corpus was constructed jointly at Lancaster University and CIIL, Mysore, India. Some of the parallel corpora exist in this corpus are English-Hindi, English-Bengali and English-Urdu (Baker et al., 2003)**.**

English-Punjabi parallel corpus: The corpus was constructed at Punjabi University, Patiala, India by (Jindal et al., 2017) for SMT system.

English-Manipuri parallel corpus: The corpus was constructed at CDAC Mumbai, India by (Singh, 2012) for SMT system.

2.2 SMT System

A large number of SMT systems have been developed globally for popular languages. Some of the SMT systems are discussed as follows:

[1]http://christos-c.com/bible
[2]https://en.wikipedia.org/wiki/Europarl_Corpus
[3]http://opus.nlpl.eu
[4]http://tdil-dc.in/index.php?lang=en

English to Arabic SMT system: The system was developed at MIT, USA by (Badr et al., 2008) using the Phrase-based SMT (PBSMT) technique and Moses. The BLEU score was 28.9.

English to Spanish SMT system: The system was developed at the University of California, Berkeley by (Nakov, 2008) using the PBSMT technique. The BLEU score was 21.92.

English to Vietnamese SMT system: The system was developed at the University of Ulsan, Ulsan, Korea by (Phuoc et al., 2016) using the PBSMT technique and Moses. The BLEU score was 32.30**.**

English⇔Welsh SMT system: The system was developed using the Phrase-based SMT approach by (Jones & Eisele, 2006) at Saarland University, Germany. The BLEU scores of the English to Welsh and Welsh to English SMT systems were 36.16 and 42.22 respectively.

French to English SMT system: The system was developed at Carnegie Mellon University, Pittsburgh, USA by (Hanneman et al., 2009) using the PBSMT approach and Moses.

Lots of SMT system has been developed for English and Indian natural languages. Some of the SMT systems are discussed as follows:

Assamese to English SMT system: The system was developed at Gauhati University, Guwahati, India by (Baruah et al., 2014) using the PBSMT approach and Moses. The BLEU score was 9.72.

English to Punjabi SMT system: The system was developed at the I. K. Gujral Punjab Technical University, Punjab, India by (Jindal et al., 2018) using the PBSMT technique.

Hindi⇔English SMT system: The system was developed at IIT Bombay, India by (Dungarwal et al., 2014) using the PBSMT technique.

Manipuri⇔English SMT system: The system was developed at Jadavpur University, Kolkata, India by (Singh & Bandyopadhyay, 2010) using the Phrase-based SMT technique and Moses.

3 Corpus Construction and Collection

The different domains of English-Bodo parallel corpus which have been used to develop the English to Bodo SMT system are discussed below.

3.1 Construction of Parallel Corpus

The construction of parallel corpus is a very laborious and difficult task. A GUI based E-BPTC (English-Bodo parallel Text Corpus) creator tool has been designed for constructing English-Bodo parallel text corpus. The tool has been designed primarily for typing the handwritten translated sentences of Bodo language of the corresponding given sentences of English language. In this tool, Unicode-based a Bodo hard keyboard and a Bodo soft keyboard have been designed for typing the texts of the Bodo language. The Bodo hard keyboard is used through the English hard keyboard. The General and News domains of English-Bodo parallel text corpus have been built using the E-BPTC creator tool. The constructed parallel corpora are discussed as follows:

General domain E-BPTC: The corpus has been constructed using the E-BPTC creator tool. The screenshot of the creator tool for building the General domain E-BPTC is shown in Figure 2.

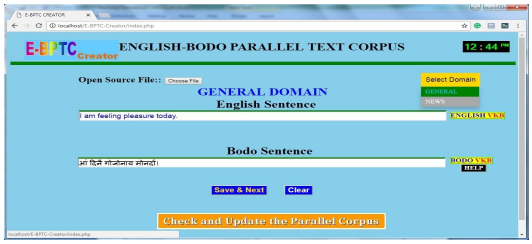


Figure 2: Screenshot of the E-BPTC creator tool for constructing the General domain E-BPTC

The corpus contains English-Bodo parallel sentences which are generally used in our daily life like communication, meeting, teaching and interview purposes. The English sentences and their corresponding translated handwritten Bodo sentences have been collected from the various sources, such as dictionaries, books, corpora and the web. Total 6,500 English-Bodo parallel sentences have been constructed in this corpus.

News domain E-BPTC: The corpus has been constructed using the E-BPTC creator tool. The screenshot of the creator tool for building the News domain E-BPTC is shown in Figure 3.

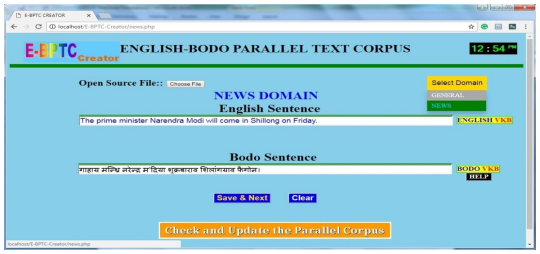


Figure 3: Screenshot of the E-BPTC creator tool for building the News domain E-BPTC

The corpus contains English-Bodo parallel sentences and the sentences have been collected mainly from the Educational news, General news, Political news and Sports news. The English sentences and their corresponding translated handwritten Bodo sentences have been collected from the different sources, such as English and Bodo Newspapers, corpora and the web. Total 4000 English-Bodo parallel sentences have been constructed in this corpus.

3.2 Collection of Parallel Corpus

The different domains of English-Bodo parallel corpus which have been collected from the TDIL programme are described as follows:

Agriculture domain E-BPTC: The corpus contains English-Bodo parallel sentences. Total 4000 English-Bodo parallel sentences have been prepared in this corpus.

Health Domain E-BPTC: The corpus contains English-Bodo parallel sentences. Total 12,300 English-Bodo parallel sentences have been prepared in this corpus.

Tourism Domain E-BPTC: The corpus contains English-Bodo parallel sentences. Total 9,200 English-Bodo parallel sentences have been prepared in this corpus.

4. English to Bodo SMT System

The English to Bodo SMT system has been developed using the Phrase-based SMT technique for the different domains of English-Bodo parallel text corpus, namely Tourism, News, Health, General and Agriculture. The PBSMT is a perfect and widely used approach of MT nowadays. It needs a huge amount of bilingual parallel aligned corpus for the best translation results

The overall architecture of the English-Bodo Phrase-based SMT system is shown in Figure 4.

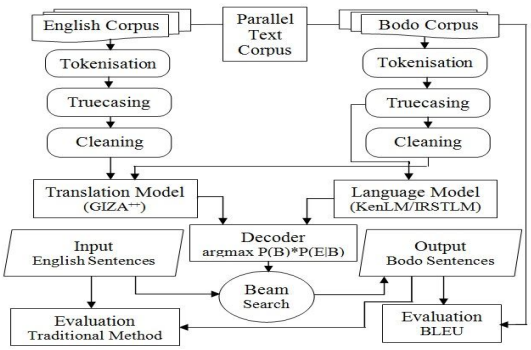


Figure 4: Overall architecture of the SMT system

The following operations have been performed for the different domains of E-BPTC to train the English to Bodo SMT system.

4.1 Corpus Preparation

Corpus pre-processing is the most essential task to prepare a bilingual parallel corpus for training the SMT system using Moses (Koehn, 2016). The following steps have been performed for the different domains of E-BPTC to build the different statistical models.

Step 1: Tokenization has been performed for the parallel corpus to insert space between the words and punctuation.

Step 2: True casing has been performed for the corpus to convert the first word of each sentence to their most probable casing.

Step 3: Cleaning has been performed to remove the long sentences, empty sentences, extra spaces and misaligned sentences from both the English and Bodo corpora.

4.2 Language Model

The Language Model (LM) is an important component of the SMT. It is built to measure the fluency of the sentences of the target language. In this system, LM has been built for the different domains of Bodo corpus using the toolkit KenLM. The LM has computed the probability of the sentences of Bodo language P(B) using the 3-gram modelling technique. It has computed the probability of a Bodo sentence as the probability of particular words P(w) using the Markov Chain Rule (Koehn, 2009) as shown in Eq. 1.

$$P(B)=P(w_1,w_2,w_3,.....,w_n)$$
$$=P(w_1)P(w_2|w_1)P(w_3|w_1w_2)P(w_4|w_1w_2w_3).....$$
$$.....P(w_n|w_1w_2...w_{n-1}) \quad (1)$$

Where w_1, w_2, w_3,……., w_n are the words of the Bodo language.

The formula for calculating tri-gram probabilities $P(w_n|w_{n-2}w_{n-1})$ of the sentences of target language is shown in Eq. 2.

$$P(w_n|w_{n-2}w_{n-1})= \frac{\text{Count}(w_{n-2}w_{n-1}w_n)}{\text{Count}(w_{n-2}w_{n-1})} \quad (2)$$

Where Count $(w_{n-2}w_{n-1}w_n)$ indicates the number of occurrences of the sequence $w_{n-2}w_{n-1}w_n$ in the corpus.

4.3 Translation Model

Translation Model (TM) is the most important component of the SMT. It is used to confirm the adequacy of the translation results in the SMT system. The TM has calculated the probabilities of the English sentence (E) and the Bodo sentence (B) based on the behaviour of the sentences, i.e. P(E|B). It can be calculated as the sum over all probabilities of all possible alignments (A) between the words or phrases in two sentences of the English and Bodo languages (Brunning, 2010) as shown in Eq. 3.

$$P(E|B) = \sum_{A} P(E, A|B) \quad (3)$$

The GIZA++ toolkit has been used in the SMT system for word or phrase alignment and to build the translation model for the different domains of E-BPTC. In the TM, a phrase translation table is created and the table ensures that the English phrases and the Bodo phrases are good translations of each other. An example of the word (or phrase) alignment in the English to Bodo Phrase-based translation model is shown in Figure 5.

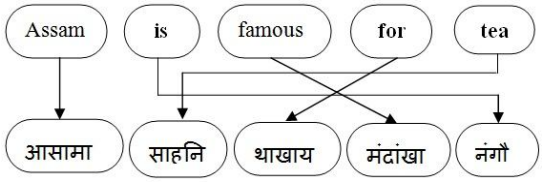


Figure 5: Word alignment between English and Bodo sentences

4.4 Decoder

The decoder is an essential component of SMT. It can find out the maximum translation probability using the output results of the LM and TM as shown in Eq. 4. The decoder uses a Beam search technique to find the best possible translation results (Koehn, 2004).

$$P(E, B) = \text{argmax } P(B) * P(E|B) \quad (4)$$

Where P(B) is the output result of the LM and P (E|B) is the output result of the TM.

5 Experimental Results

The English to Bodo SMT system has been tested several times using the different numbers of English-Bodo parallel sentences for the different domains of E-BPTC, namely Tourism, News, Health, General and Agriculture. It has achieved good translation results in the SMT system using the General domain E-BPTC. The English-Bodo parallel sentences which have been used to train, tune and test the SMT system for the different domains of E-BPTC are shown in Table 1.

Multi-domain E-BPTC	Languages	Training		Tuning	Testing
		Words	English-Bodo parallel sentences		
Agriculture	English	85,901	4,000	500	3,000
	Bodo	64,935			
General	English	52,778	6,500	600	5,000
	Bodo	41,920			
Health	English	224,077	12,300	1000	10,000
	Bodo	172,723			
News	English	45,914	4,000	500	3,000
	Bodo	38,205			
Tourism	English	199,570	9,200	1000	8,000
	Bodo	165, 214			

Table 1: No. of sentences used for training, tuning and testing the SMT system

6 Evaluation

The Translation Accuracy (TA) of the different domains of English-Bodo parallel corpus in the English to Bodo SMT system has been evaluated using the Manual or Human and Automatic or Machine evaluation techniques.

6.1 Manual Evaluation Technique

The translation accuracy of the different domains of E-BPTC in the SMT system has been evaluated by a linguistic person Dr. Ismail Hussain, Assistant Professor, Dept. of Bodo, Gauhati University, Guwahati, India. He has evaluated the TA (in terms of percentage) based on the adequacy and fluency of the input English sentences and their corresponding translated or output Bodo sentences. The levels of TA of the different domains of parallel corpus in the SMT system are shown in Table 2.

Levels	Definitions	TA (%) of the multi-domain parallel corpora				
		Agriculture	General	Health	News	Tourism
Perfect	A	35	45	40	40	38
Fair	B	32	36	34	37	35
Acceptable	C	25	14	20	18	20
Nonsense	D	8	5	6	5	7

Table 2: Levels of TA of the multi-domain E-BPTC (Approx)

In the above table, the definition A means Translated Bodo sentences are very good to understand, B means Translated Bodo sentences are easy to understand but need a minor correction, C means Translated Bodo sentences are broken but are understandable, and D means Translated sentences are not understandable.

6.2 Automatic Evaluation Technique

The translation accuracy of the different domains of E-BPTC in the SMT system has been evaluated using the BLEU (Bilingual Evaluation Understudy) technique. The BLEU is a popular automatic and language independent evaluation technique. It can evaluate the best translation accuracy in an SMT system. The BLEU score is computed based on the average of matching n-grams between a proposed or candidate translation (in this case, Machine translated Bodo corpus) and a reference or human translation (in this case, human translated Bodo corpus). The BLEU score seems to correspond well with the human judgment based on the fluency and accuracy (Uszkoreit, 2007). The BLEU scores of the different domains of E-BPTC are shown in Table 3.

Multi-domain E-BPTC	Translation	Bodo Sentences	BLEU Scores
Agriculture	Reference	3000	29.25
	Candidate		
General	Reference	5000	38.12
	Candidate		
Health	Reference	10000	36.05
	Candidate		
News	Reference	3000	32.96
	Candidate		
Tourism	Reference	8000	35.40
	Candidate		

Table 3: BLEU scores of the different domains of E-BPTC

7 Conclusion and Future Work

In this paper, the English to Bodo SMT system has been developed using the Phrase-based SMT technique for the different domains of English-Bodo parallel corpus, namely Tourism, News, Health, General and Agriculture. A GUI based E-BPTC creator tool has been developed for building the General and News domains of English-Bodo parallel text corpus. The translation accuracy of the different domains of E-BPTC has been evaluated using the Manual evaluation and BLEU techniques. The General domain E-BPTC has produced good translation results in the English to Bodo SMT system.

The SMT system can be extended by adding more number of good quality parallel sentences in the different domains of English-Bodo parallel corpus to achieve the best translation results. The accuracy of the translation results of the different domains of E-BPTC can be enhanced using the Machine transliteration technique in the SMT system. The research work can be explored by developing bidirectional English⇔Bodo MT system using the NMT approach for the different domains of English-Bodo parallel corpus like Agriculture, General, Health, News and Tourism.

References

Antony P. J. 2013. Machine Translation Approaches and Survey for Indian Languages. *Computational Linguistics and Chinese Language Processing (CLCLP)*, 18(1):47-78.

Ayana Kuandykova, Amandyk Kartbayev, and Tannur Kaldybekov. 2014. English-Kazakh Parallel Corpus for Statistical Machine Translation. *International Journal on Natural Language Computing*, 3(3): 65-72.

Dafydd Jones and Andreas Eisele. 2006. Phrase-based Statistical Machine Translation between English and Welsh. *In the proceedings of the 5th SALTMIL workshop on Minority Languages*, pp.75-77.

Greg Hanneman, Vamshi Ambati, Jonathan H. Clark, Alok Parlikar, and Alon Lavie. 2009. An Improved Statistical Transfer System for French-English Machine Translation. *In the Proceedings of the 4th Workshop on Statistical Machine Translation*, ACL, Athens, Greece, pp.140-144.

Hans Uszkoreit. 2007. Survey of Machine Translation Evaluation. *EuroMatrix Project*, Saarland University, Germany, pp.1-80.

Ibrahim Badr, Rabih Zbib, and James Glass. 2008. Segmentation for English-to-Arabic Statistical Machine Translation. *In the Proceedings of Association for Computational Linguistics-08: HLT*, USA, pp.153-156.

James Brunning. 2010. Alignment Models Algorithms for Statistical Machine Translation (PhD Thesis). Cambridge University, UK.

Kalyanee K. Baruah, Pranjal Das, Abdul Hannan, Shikhar kr. Sarma. 2014. Assamese-English Bilingual Machine Translation. *International Journal on Natural Language Computing (IJNLC)*, 3(3): 73-82.

Kathiravan, P., Makila, S., Prasanna, H., and Vimala, P. 2016. Overview- The Machine Translation in NLP. *International Journal for Science and Advanced Research in Technology*, 2(7):19-25.

Liang Tian, Derek F. Wong, Lidia S. Chao, Paulo Quaresma, Francisco Oliveira, and Lu Yi. 2014. UM-Corpus: A Large English-Chinese Parallel Corpus for Statistical Machine Translation. *In the proceedings of the 9th International Conference on Language Resources and Evaluation.* European Language Resources Association (ELRA), pp.1837-1842.

Nguyen Quang Phuoc, Yingxiu Quan, and Cheol-Young Ock. 2016. Building a Bidirectional English-Vietnamese Statistical Machine Translation System by using Moses. *International Journal of Computer and Electrical Engineering*, 8(2):161-168.

Paul Baker, Andrew Hardie, Tony McEnery, and B.D. Jayaram. 2003. Constructing Corpora of South Asian Languages. *UK EPSRC-Project, Department of Linguistics, Lancaster University and Central Institute of Indian Languages*, Mysore, India, pp.71-80.

Philipp Koehn. 2004. Pharaoh: A Beam Search Decoder for Statistical Machine Translation. *In the proceedings of the 6th Conference of the Association for Machine Translation in the Americas*(AMTA), Springer.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. *In the proceedings of the MT Summit X.*

Philipp Koehn. 2009. Statistical Machine Translation (e-Book). *Cambridge University Press*, New York, pp.1-447.

Philipp Koehn. 2016. MOSES (User Manual). Statistical Machine Translation (SMT) system, *University of Edinburgh*, UK.

Piyush Dungarwal, Rajen Chatterjee, Abhijit Mishra, Anoop Kunchukuttan, Ritesh Shah, and Pushpak Bhattacharyya. 2014. The IIT Bombay Hindi⇔English Translation System at WMT 2014. *In the proceedings of the Ninth Workshop on Statistical Machine Translation*, ACL, Maryland, USA, pp.90-96.

Preslav Nakov. 2008. Improving English-Spanish Statistical Machine Translation: Experiments in Domain Adaptation, Sentence Paraphrasing, Tokenization, and Recasing. *In the proceedings of the 3rd Workshop on Statistical Machine Translation*, Columbus, USA, pp.147-150.

Saiful Islam, Maibam Indika Devi, and Bipul Syam Purkayastha. 2017. A Study on Various Applications of NLP Developed for North-East Languages. *International Journal on Computer Science and Engineering (IJCSE)*, 9(6):368-378.

Saiful Islam and Bipul Syam Purkayastha. 2018. English to Bodo Phrase-Based Statistical Machine Translation. *In the proceedings of the 10th International Conference on Advanced Computing and Communication Technologies (ICACCT-2017). Advances in Intelligent*

Systems and Computing, Springer, Singapore, vol. 562: 207-217.

Shishpal Jindal, Vishal Goyal, and Jaskarn Singh Bhullar. 2017. Building English-Punjabi Parallel Corpus for Machine Translation. *International Journal of Computer Applications*, 180(8):26-29.

Shishpal Jindal, Vishal Goyal, and Jaskarn Singh Bhullar. 2018. English to Punjabi statistical machine translation using Moses (Corpus-Based). *Journal of Statistics and Management Systems*, 21(4):553-560.

Thoudam Doren Singh and Sivaji Bandyopadhyay. 2010. Manipuri-English Bidirectional Statistical Machine Translation Systems using Morphology and Dependency Relations. *In the proceedings of SSST-4, Fourth Workshop on Syntax and Structure in Statistical Translation*, COLING-2010, Beijing, pp.83-91.

Thoudam Doren Singh. 2012. Building Parallel Corpora for SMT System: A Case Study of English-Manipuri. *International Journal of Computer Applications (IJCA)*, 52(14): 47-51.

POS Tagging and Named Entity Recognition on Handwritten Documents

Vijay Rowtula, Praveen Krishnan, C.V. Jawahar
CVIT, IIIT Hyderabad
{vijay.rowtula, praveen.krishnan}@research.iiit.ac.in and jawahar@iiit.ac.in

Abstract

Parts of Speech (POS) tagging and Named Entity Recognition (NER) on handwritten document images can help in keyword detection during document image processing. In this paper, we propose an approach to detect POS and Named Entity tags directly from offline handwritten document images without explicit character/word recognition. We observed that POS tagging on handwritten text sequences increases the predictability of named entities and also brings a linguistic aspect to handwritten document analysis. As a pre-processing step, the document image is binarized and segmented into word images. The proposed approach comprising of a CNN-LSTM model, trained on word image sequences produces encouraging results on challenging IAM dataset.

1 Introduction

Information extraction from handwritten document images has numerous applications, especially in digitization of archived handwritten documents, assessing patient medical records and automated evaluation of student handwritten assessments, to mention a few. Document categorization and targeted information extraction from various such sources can help in designing better search and retrieval systems for handwritten document images. Keyword spotting (Fischer et al., 2012) is used for automatic document categorization by detecting the keywords or named entities directly on handwritten document images rather than transcribing to text to find keywords.

Semantic annotation of handwritten documents, especially spotting keywords using POS tags or NER is relatively a newer problem with very few works emerging on this front. In this paper, we attempt to fill the gap by proposing an approach for POS tagging and NER without handwriting transcription. The contribution of this work is to show generalization with a similar or improved performance of a unified end-to-end model without separating the sequence of sub-processes involved, thereby avoiding error propagation. Identifying named entities using noun phrases from POS tags can also be greatly helpful for keyword-based document retrieval. Detecting named entities irrespective of its structural and positional characteristics (eg. uppercase or lowercase letters) is an advantage of our approach. As a pre-processing step, we choose a handwritten dataset with segmented words and POS tag annotations. It helped us focus only on the aspect of POS and named entity tagging on handwritten word images rather than the problem of word segmentation from handwritten documents.

Related Works: Several state-of-the-art NER techniques were published in the literature using handcrafted features (Ritter et al., 2011; Lample et al., 2016). Transcription based models such as (Romero and Sánchez, 2013; Prasad et al., 2018; Carbonell et al., 2018) trained Handwritten Text Recognition (HTR) and NER jointly, to mitigate the disadvantage of errors in the first module affecting the next. But in historical handwritten documents, handwriting recognition struggles to produce an accurate transcription thereby reducing the accuracy of the whole system. Adak et al. (Adak et al., 2016) described an approach to directly detect the named entities from the document images. They used handcrafted features from document images with LSTM classifier, thereby avoiding the transcription step. The method relies on handcrafted features like identifying capital letters to detect possible named entities.

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 87–91

Figure 1: Example of POS and NE tagging on a sentence chosen from IAM handwritten dataset.

2 Our Approach

We hypothesize that, with sufficient handwritten document data and pre-processing, a deep learning model will be able to predict POS tags and named entities despite the inherent complexity, without the need for transcription.

2.1 Direct learning using synthetic dataset

Deep learning architectures need large datasets to attain decent results on image recognition tasks and finding sufficient handwritten document images is a challenging task. Hence we first trained the model on synthetic handwritten word images. We used a standard parts-of-speech dataset to create a synthetic handwritten dataset using artificial fonts, as described in (Krishnan and Jawahar, 2016). We used the same font for each sentence and sufficient data augmentation in the form of noise, translation, and rotation to resemble a large real handwritten dataset. Our assumption is that, with sufficient data, a deep learning model can generalize well on the end-to-end task without breaking it into sub-tasks (Liu et al., 2016). For POS tagging on handwritten text, our first step was to choose a model trained on word spotting in handwritten document images. The use of deep learning architectures to capture spatial features of word images is widely discussed in (Krishnan and Jawahar, 2016; Krishnan et al., 2016). The authors used HWNet architecture trained on 1 million word image dataset to make it robust to most handwriting variations. We initially used the pre-trained model (HWNet) to extract the features of synthetic handwritten words and, later fine-tuned a separate neural net on these features to classify POS tags. We considered this model was our baseline for the best performance that can be achieved using a pre-trained model on handwritten word images. In our alternate training scheme, we directly train a deep model on word images to classify POS tags. We observed that the model performance was similar to HWNet feature-based model which affirmed our assumption that translation into text or feature extraction sub-tasks may not be required for POS tagging on handwritten word images.

2.2 POS Tagging and NER

The model trained on the synthetic dataset is fine-tuned on a real handwritten dataset. We tested various architectures (CNN, CNN-LSTM) for both POS tagging and NER on a challenging handwritten document dataset. Some of them are discussed below.

Deep CNN model for POS tagging: Convolutional Neural Nets (CNN) are good in capturing the intricate details of images, hence making the model stable to inconsistencies like noise and translation (Krizhevsky et al., 2012). We trained a ResNet (He et al., 2016) model with 35 layers (validated empirically) on the synthetic dataset and fine-tuned it on IAM dataset for POS tagging task. The ResNet-35 ends with a softmax layer that outputs the probability distribution over the class la-

bels (POS tags). We trained the model with cross-entropy loss function to predict the class labels.

CNN-LSTM model for POS tagging: The probability of a transition between words may depend not only on the current observation, but also on past and future observations, if available (Lafferty et al., 2001). Since sentences in handwritten document images are word image sequences, we next used a combination of ResNet (CNN) and LSTM layers for training a POS tagging model on sequential information. We appended two layers of LSTM after ResNet-35 blocks and converted the input to LSTM as time distributed sequence. Different sequence lengths were tested on POS tags (classes). We report the performance of changing sequence lengths in the results section.

Named Entity Recognition: We adapt the similar architectures (CNN, CNN+LSTM) for the problem of NER. Here the underlying CNN architecture is ResNet-35. However, neither of the models had higher accuracy as noticed in similar experiments reported in (Toledo et al., 2016). We observed that named entities are related to position and distribution of POS tags in a sentence. We trained a multi-output classification network with architecture similar to POS model, with an extra branch of dense layers from the first fully connected dense layer, for named entity prediction. Hence the model now has an independent output with loss calculated from two sets of classes. As described in section 3.1, named entities have class imbalance problem. This is one of the reasons for choosing outputs separated by multiple dense layers rather than a common layer training for multi-class classification. We initially trained the network simultaneously for both POS and NER. We observed that though POS prediction accuracy remained the same as independent POS training, NER training did not give encouraging results. Hence we first trained the model (ResNet + LSTM + dense layers) for POS tagging by freezing the dense layers of NER. After the network achieved satisfactory accuracy on POS tagging, we froze the POS part of the network - including the ResNet-LSTM layers and trained just the dense layers of NER. We used altered class weights to tackle the class imbalance problem. This method improved the accuracy of NER better than any of the methods we have tried earlier.

Named Entities	**Tags**
Date	DATE
Geopolitical Entity	GPE
Organization	ORG
Person Name	PERSON
Nationalities or Religious or Political Groups	NORP
Unrelated	OTHERS
Not an Entity	–

Table 1: Named Entities used for our analysis.

3 Experimental Results and Discussions

Dataset: We used two different datasets, for training and fine-tuning the models. For training, a synthetic handwritten dataset was generated from chunking dataset of CoNLL-2000 shared task (Tjong Kim Sang and Buchholz, 2000), randomly using some of the 100 publicly available handwritten fonts (Krishnan and Jawahar, 2016). The chunking dataset contains sentences aligned with 211727 text tokens along with their POS tags in a separate train and test text files. This dataset was initially used for training and validation. The model is further fine-tuned on IAM handwritten dataset (Marti and Bunke, 2002). The IAM dataset contains 1539 forms written by 657 authors. The forms are further segmented into 115320 words and are annotated with POS tags. Though IAM dataset contains segmented lines and sentences, they are not properly annotated with text accordingly which makes it difficult to demarcate the individual sentences accurately. Hence we separated sentences based on pre-defined sentence rules based on words and cross-validated them using python based NLP tool named "Spacy".

Since the IAM handwritten forms have transcripts, the text was fed into the Spacy for generating the ground truth named entities. Spacy tagged sentences with 17 different categories of named entities. Though we restricted the classes to 6 named entities by choosing most recurrent tags, there was a class-imbalance problem. The list of tags used in this work is shown in Table 1. The unrelated entities occupied 92% of the NER classes. The IAM dataset is available as train, validation1, validation2, and test partitions. We used the training set to fine-tune our models and validated them against validation1 and validation2 sets.

Experiments	Precision	Recall	F1-score
Neural Net trained on HWNet features - CoNLL-2000 dataset synthetic images (POS tagging).	92.4	87.2	89.7
ResNet trained on - CoNLL-2000 dataset synthetic images (POS tagging).	94.2	84.5	89
ResNet trained on - CoNLL-2000 dataset synthetic images and fine-tuned on IAM dataset (POS tagging).	75.4	64.8	69.7
ResNet + LSTM trained on - CoNLL-2000 dataset synthetic images and fine-tuned on IAM dataset (POS tagging).	76.2	66.8	**71.2**
ResNet + LSTM trained on - CoNLL-2000 dataset synthetic images and fine-tuned on IAM dataset (NER).	74	64.1	**68.7**

Table 2: List of conducted experiments with precision, recall and F1-scores.

3.1 Results and Discussion

As a baseline on the synthetic dataset, we initially extracted HWNet features on word images from the fully connected layer and trained a multi-layered perceptron on 36 POS classes provided by CoNLL-2000 dataset. The model achieved an F1-score of 89.7. We then trained a 35 layer ResNet model which achieved an F1-score of 89. This was our initial experiment to prove that a model can be trained to classify POS tags directly on handwritten word images, rather than a feature based model training.

POS tagging on IAM dataset: The ResNet model trained and validated on the synthetic CoNLL-2000 dataset is fined tuned on IAM dataset. We initially trained directly on word images to classify 58 POS tags without the sequence information. The architecture essentially contained no LSTM layers. The ResNet model achieved an F1-score of 69.7 on IAM test dataset. We altered the architecture and dataset to include sequence information. We replaced dense layers succeeding the CNN layers with LSTM layers and trained the model with varying sequence lengths of 3, 64, 128 and 256 words. We observed that ResNet-LSTM model trained on 128 word length sequences performed best with an F1-score of 71.2 We attribute the decline of prediction accuracy on IAM dataset compared to synthetic dataset due to the following reasons. (i) Distortions in word images - We observed that most of the word images are formed by concatenating individual characters. (ii) Character distortions - characters such as '.' and ',' are displayed as 'l' in the dataset. (iii) Proper nouns errors - proper nouns do not start with capital letters. We also observed that 26% of errors were due to the noun form of words (NN), followed by adjectives (JJ) at 18% and conjunctions (IN, TO) at 12%. Rest of the errors were due to special characters, commas, and full stops.

NER on IAM dataset: Our models, training methods and metrics are summarized in Table 2. We used class weights to bias the training towards named entity tags other than "unrelated" class, to handle class imbalance problem. We initially trained IAM dataset words for two tasks in parallel using the architecture described in Section 2.2. But the accuracy of such model was low on NER task. Our first observation was that the errors caused by class imbalance were propagated back to the complete model which impacted the performance of both POS tagging and NER as well. Hence we first trained the model on POS tagging by freezing the NER layers, then we froze the layers for POS tagging and trained the model on NER. After 20 epochs, we fine-tuned the whole model further using very low learning rate for 10 epochs. The ResNet-LSTM model gave F1-score of 68.7 on NER on handwritten text.

4 Conclusion

A POS tagger and named entity recognizer for offline handwritten unstructured documents, without employing a character/word recognizer and an independent linguistic model, is presented in this paper. Experiments conducted on IAM dataset have resulted in an average F1-score of 71% on POS tagging and 68% on NER task. The proposed method is expected to work in other languages as well since our method deals with the linguistic aspect of handwritten documents where POS tags are identified first and then the NER.

References

Chandranath Adak, Bidyut B Chaudhuri, and Michael Blumenstein. 2016. Named entity recognition from unstructured handwritten document images. In *Document Analysis Systems (DAS), 2016 12th IAPR Workshop on*. IEEE.

Manuel Carbonell, Mauricio Villegas, Alicia Fornés, and Josep Lladós. 2018. Joint recognition of handwritten text and named entities with a neural end-to-end model. *arXiv preprint arXiv:1803.06252*.

Andreas Fischer, Andreas Keller, Volkmar Frinken, and Horst Bunke. 2012. Lexicon-free handwritten word spotting using character hmms. *Pattern Recognition Letters*, 33(7).

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*.

Praveen Krishnan, Kartik Dutta, and CV Jawahar. 2016. Deep feature embedding for accurate recognition and retrieval of handwritten text. In *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on*. IEEE.

Praveen Krishnan and CV Jawahar. 2016. Matching handwritten document images. In *European Conference on Computer Vision*. Springer.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Hao Liu, Jiashi Feng, Meibin Qi, Jianguo Jiang, and Shuicheng Yan. 2016. End-to-end comparative attention networks for person re-identification. *arXiv preprint arXiv:1606.04404*.

U-V Marti and Horst Bunke. 2002. The iam-database: an english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1).

Animesh Prasad, Hervé Déjean, Jean-Luc Meunier, Max Weidemann, Johannes Michael, and Gundram Leifert. 2018. Bench-marking information extraction in semi-structured historical handwritten records. *arXiv preprint arXiv:1807.06270*.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Verónica Romero and Joan Andreu Sánchez. 2013. Category-based language models for handwriting recognition of marriage license books. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*. IEEE.

Erik F Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*. Association for Computational Linguistics.

J Ignacio Toledo, Sebastian Sudholt, Alicia Fornés, Jordi Cucurull, Gernot A Fink, and Josep Lladós. 2016. Handwritten word image categorization with convolutional neural networks and spatial pyramid pooling. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer.

A New Chat Solution for Shared Services using Natural Language Processing Models

M. Saravanan
Ericsson Research
Chennai, India
m.saravanan@ericsson.com

Satheesh K. Perepu
Ericsson Research
Chennai, India
perepu.satheesh.kumar@ericsson.com

Sudipta Bose
Dept. Computer Science & Engg
IIT Dhanbad, India
sudiptabose94@gmail.com

Abstract

In the service industry, the shared services denote an accountable entity which started taking the lead over managed services for the past three decades. In shared services, resources are shared across different engagements or projects thereby making them cost-efficient as they centralize back-office operations that are used by multiple divisions of the same company and eliminate redundancy. Quite often in shared services, people face issues with addressing multiple queries. In this paper, we have designed a new chat solution named as **ECHAT** which can answer the complex queries raised by the service engineers related to the product by employing Knowledge Graph and Deep Learning Algorithm. EChat would engage with the service engineers in natural language and would be available as any messaging platform to induce automation in the process. Related to this, we have introduced a new framework which explores different Natural Language Processing (NLP) methods to consider the position of the word, its relatedness and to generate random vector representation to avoid the false positives in responses. Moreover, the significance of our framework is that it can be implemented in any new environment with the available dataset to endure fast training and give more relevant and accurate answer to user queries. EChat has been evaluated and found that the accuracy in providing correct responses for the complex user queries will reach 91% on an average.

1 Introduction

Shared services represent business operations that are handled by multiple parties in an organization. Mainly there are two rationales for shared services set up in an industrial environment i) Less of a common resource ii) Improve Efficiency. Nowadays the industries are very cautious in handling the high operational cost, by allotting the only minimum required number of managers to run the departments with shared resources and also to maintain the efficiency through industrialization which is based on specialization and standardization. Shared services are more than just central organization or consolidation of similar activities in a location. It involves running service activities like a business deliverable for internal customers at a cost, quality and timeliness that is competitive with alternatives.

Recently by introducing automation through digitalization, most of the companies prefer to use chat application to handle the customer conversations with the help of a machine which are called ChatBots. Chatbots are expected to understand a user's query and deliver prompt answers to solve the customer's issues on a real-time basis. Regarding this, a chat interface is suitably designed to allow a bot to converse with multiples users in a simple, fastest and easiest way as possible. It clearly necessitates the need for chatbot to establish automatic conversation in the industry for shared service operations to speed up the present communication channel.

A chatbot is essentially a digital employee that can answer simple customer service questions autonomously in the present digital world. It is generally categorized into two types: Command-based and AI-based. Command-based chatbot relies on a newly constructed database of replies and the relevant heuristics. The bots reply in a way by selecting an answer that matches the context of a query. It is not trained to extend for the creation of new texts and hence it can answer only a limited set of questions. In general, command-based chatbot can perform on its own limitations. On the other hand, AI-based or Machine Learning (ML)-based chatbots gives replies based on training and applying NLP techniques for understanding and interpreting the texts. These chatbots become smarter with time, learning from past questions and answers. Chatbots used for different purposes are typically limited to conversations regarding a specialized purpose and not for the entire range of human communication.

Based on the specific requirements in shared services, we have designed a new chat framework named as EChat which follows AI-based

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 92–96

implementations for service engineer's conversation with developers relate to the product issues. It explores different NLP methods and ML techniques in introducing efficient custom-built chatbot to answer most of the customer queries and only very few are forwarded to human agent to exhibit details with the profundity of the issue based on the user requirement. We have evaluated the performances of EChat by implementing in a specific shared service of our company for managing a single product query.

2 Related Works

Shared services in any organization relate to the idea of sharing resources and communication within an organization or group. The goal of a shared services delivery model is to allow each business division to focus its limited resources on activities that support the division's business goals. In this paper, we have introduced a chat solution (EChat) for establishing 24*7 communication with service engineers. The chat interface design is the process that offers credible interaction between the user and the machine.

For texting dialects and SMS writing, Choudhury et al (2007) proposed a Hidden Markov Model for normalizing the text. A statistical classifier based normalization for text messages was introduced by Pennell and Liu (2010). These methods were used to study which character to delete and when to normalize the text by reversing the representations. Xi et al (2004) utilize a positioning function to choose the most applicable messages to client questions in newsgroup searches, but, in which the author feature demonstration is not feasible. An Enhanced language-independent approach of conversational agent for question answering using semantic web knowledge has been developed by Alexandru Dobrila (2010). Cui et al (2015) proposed to constrain morphologically similar words to have similar representations. Soricut and Och (2015) described a method to learn vector representations of morphological transformations, allowing to obtain representations for untrained words by applying pre-determined rules. Nishimura et al (2005) build up a learning base for a question-answering framework that answers compose 'how' questions. From these studies, we understand the feasibility and impact of applying the different statistical technique for processing the text in our proposed framework.

A popular language for creating chatbot is the Artificial Intelligence Markup Language (AIML). With the use of AIML, one can program a computer to give a specified set of answers to a specified set of questions. Thomas and Amrita (2016) build up an AIML and LSA based chatbot to solve the client requirements in managing E-business sites. Shahriare and Shamim (2015) demonstrated the audit of utilization of the chatbot which are created utilizing the AIML contents. It is very clear that AIML based chatbots are lightweight and productive to any work environment. We have used AIML in our proposed chat framework.

Vinyals and Le (2015) proposed a recurrent neural network sequence generation based chatbot to generate the optimal response. For text to speech systems, Sproat and Jaitly, (2016) proposed different Recurrent Neural Network architectures to normalize texts to correctly spoken form. Duplessis et al (2016) built a new chat application named as ChatterBot. The chatterbot focuses on the local coherence of dialogue. Indeed, it only takes into account the last user utterance to select its response. Since this application is very close to our approach, we have tried to compare our proposed chatbot with the chatterbot.

After considering the advantages and disadvantages, we have constructed a new AI-based chatbot using AIML by demystifying Deep Learning techniques in a new framework under the conversation-oriented platform for efficient handling of extracted information.

3 EChat Framework and Models

We have built a new chat application and named it as **EChat.** It is entrusted in an intelligent way of question answering in shared service tasks. Users can enter their questions in the text area of user interface and our system processes the question and responds to the user's queries with suitable resolutions/solutions. Some cases, it rationalizes the selection with a few additional questions.

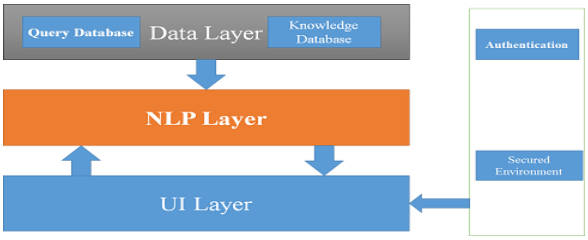


Figure 1. EChat Framework

The proposed framework of EChat is shown in Fig 1. It has three layers (i) User Interface (UI) layer (ii) Natural Language Processing (NLP) layer which performs all the needed text computation and (iii) data layer, where the historical data is available in the form of database.

The UI layer is significant to establish meaningful communication with users. It acts as the user interface between humans who are interacting to get the details using UI and NLP layer where the backend processing of texts happens. The UI layer is written in angular java script (angular JS), which sends the data to NLP layer written in Python. It can also acquire the data generated by the backend to display in UI. To interface the angular JS with python, we have used flask server (Miguel 2014), as it is an efficient way of doing the integration.

The NLP layer forms the backend of the EChat. This layer utilizes the data already available in the form of queries and solutions from the database to answer the questions entered by the service Engineers. To answer the queries not present in the database, we utilize the text in other resources of the product.

To identify the context of the query, we have applied three different models (i) Support Vector Machines (SVM), (ii) Convolutional Neural Network (CNN) and (iii) Conditional Random Field (CRF). The above models were trained through the available dataset. Further, we used the above mentioned trained models to classify the context. As the size of the training dataset is small, the models built resulted in poor accuracy generation. Hence, we prefer to go with a model free approach to solve the problem.

Our approach is explained through two phases (i) training phase and (ii) testing phase. In training, we use the dataset available to generate features. In testing, we use the trained features and user input query to give the required solution.

For training, we performed the following operations on the dataset. First, we extract all the unique words from all the sentences and find out the position of the words in each sentence. Using the position of the words in each sentence as row and unique words as column we form a matrix that provides a useful information on the number of times a particular word present in a specific position. Next step is to assign a random number of 10 dimensions for each unique word in the data. Here, we have to multiply the obtained random number of each word with the corresponding position of the word and further sum up the values to obtain 10-dimension vector for each sentence in the dataset. Finally, we normalize the value obtained to standardize the values between to 0 to 1. In the end, we obtain a ten-dimensional vector for each query available in the database.

To test the chatbot designed, we first follow the steps by considering the importance of processing query sentences.

a. Understand the importance of conjunctions present in the query sentence.

b. To handle the case of the user query similar to the one in the dataset but with different vocabulary, we used a thresholding mechanism to transform the query to that one available in the dataset. For that, we apply the Euclidean distance measure to calculate query vector with possible queries available in the training phase. Here, if the distance between vectors is low, then it means that the query gets the exact match in the database. In this case, we can give the user a solution for the matching query in the database without any additional process. If the distance obtained is lower than a specific threshold, then we will ask the user to provide more information relevant to query words by questioning them with categorical questions. This can help in establishing a conversation-oriented platform in our chat application.

c. To handle large distance values, we use trained CNN model from the knowledge graph to identify the context and pass on the solution to the user. Here we have used the other documentation available related to the product to pick the relevant answers. It the end, if both the cases failed to provide an answer then the specific question will be redirected to concerned service engineer to answer the query.

As discussed in the introduction, the proposed EChat built with intelligence system which helps users to get answers for some queries which are not available in the dataset used to train the model. For this, we consider the documentation of the product available in the knowledge database in the data layer and any other supporting documents belong to the product. First, we preprocess all the data by removing the stop words, punctuation etc. as with any traditional NLP exercise. Further, we extract the terms which are categorized as nouns and verbs by the inbuilt gensim [Radim and Petr, 2010] categorizer. In addition, we also consider the words which are important for the product, as there are chances that these words are not categorized as verbs or nouns in the general dictionary. These

words are manually labeled as nouns or verbs to use them in the exercise. As the next step, we collect instances where these words are present in the system and create knowledge graph by connecting the nouns in the system. The knowledge graph relates the relation between the keywords in the form of the entire sentence.

Hence, to provide an accurate answer we need a model to identify the context of the query. It should be noted that the knowledge graph, in general, can have many nodes (as many as the nouns in the text) and also many edges (as many as the verbs in the text).

Finally, for the initial testing, we have been provided with a text database for one of the product as a sample set. As discussed, we have generated two databases relevant to text processing. (i) Query database and (ii) Extracted Knowledge database. As part of product usage, customers come up with their queries relevant for day-to-day functioning and the engineers should provide the solutions to persuade them. These conversations happened in the form of e-mail conversations. As part of a single query, it is possible to have two or more mail chains associated with this. This is because the engineers need more details from the customers related to their problem for giving an efficient solution. To create a database of the queries and solutions, we collect the first query mailed by the customer and the relevant solution proposed by the engineer. Also, we collect some other details as a category of the query, the time is taken to provide the solution etc. To tackle the non-availability of queries in the query database, we have generated a knowledge database by collecting the details from documentation and frequently answered questions of the product. In addition, we also collected documentation of the open source tools used for the development of the product for the efficient answering of the queries which are not present in the query database.

4 Evaluation and Discussion

There are many evaluation metrics available to measure the chatbot performances. One such important metric is accuracy. Accuracy is measured in terms of how many times the chatbot provides the correct answer for the user queries. In addition to the accuracy, we also considered precision and recall measures, since we have used the classification model to understand the context present in the query. Here, precision is a percentage of relevant documents retrieved and recall is the percentage of retrieved relevant documents. We have calculated the discussed measures for both EChat and chatterbot and the details are provided in two different tables.

Table I: Accuracy Measurement

Query/Statement	Chatterbot Accuracy	EChat
Direct statement	100%	100%
Changing the position Of the statement	40%	93.33%
Combined two statement to form a single statement	45.83%	87.5%
Partial sentence as statement	66.6%	83.33%

The performance of chatbots is evaluated based on the type of questions user queried to understand the effectiveness of the proposed EChat. We have evaluated EChat performances for three different types of queries (i) direct query, (ii) partial query and (iii) combined query (two queries in the database combined). The same exercise will be done with chatterbot, a chatbot considered for comparison. The accuracies obtained for three different cases for both chatbots are reported in Table I. First case i.e. when the query is direct statement i.e. user's query exactly matches the question stored in dataset then the proposed EChat and also the chatterbot delivers 100% accuracy. Further to the second case i.e. when the user enters a partial query, the chatterbot fails to provide better results as chatterbot searches for the closest match known statement to the query available in the database. However, the EChat generates better results due to the consideration of the position of the words in a query. Moreover, in the third case when the user combines two questions or statements from the dataset to form one single statement and pass it as a query. In this case, the system should return the answer to the question which forms the larger part of the combined statement. In the third case also our approach gives the better accuracy compared to chatterbot due to an intelligent implementation of the knowledge graph.

Table II: Average Precision and Recall Score

Evaluation	EChat bot	Chatterbot
Avg. precision score	76.92%	61%
Avg. recall score	89.42%	78.52%

As already discussed, precision and recall values are calculated for three different queries as discussed. For each type, we calculate the precision,

recall and the values are averaged to obtain average precision and average recall for all the different types and the values are given in Table II. From the results, it is evident that the proposed chatbot performs exceedingly well compare to chatterbot in terms of precision and recall. In addition to the above measures, we have also tested our EChat which can deliver answers even if the queries are not similar to the queries present in the query dataset. In the domain testing, we found that 90% of times EChat returned correct answer using the words extracted from the documentation of the product. The accuracy can be improved if we use additional documents in addition to the documentation of the product. For this case, we skip the comparison as the chatterbot is not designed for the same.

5 Conclusion and Future work

We have developed EChat for shared services in industrial set up to address the queries raised by the service Engineers. We trained our dataset using query importance, sentence selection and keyword based probabilistic and knowledge-based models. Compared with the conventional chatbots like chatterbot, the proposed EChat has the strong ability to give more accurate and relevant answer to the user's questions and also work with the human agent to handle inquiries that cannot be resolved programmatically. It also delivered good performance and delivered answers with higher accuracy. Further, to answer queries which are not present in the query database, we used the knowledge graph constructed from additional documents to supply answers. Future directions include automatic regeneration of answers in our conversational chatbot which provides the best experience to the users.

References

http://www.atkearney.in/operations/ideas-insights/article/-/asset_publisher/LCcgOeS4t85g/content/rewriting-india-s-shared-services-

Reshmi, S., &Balakrishnan, K. 2016. Implementation of an inquisitive ChatBot for database supported knowledge bases. Sādhanā, 41(10), 1173- 1178.

Vinyals, O. and Le, Q. 2015. A neural conversational model. ICML Deep Learning Workshop.

Thomas N. T., Amrita Vishwa. 2016. "An E-business ChatBot using AIML and LSA", Intl. Conference on Advances in Computing, Communications and Informatics (ICACCI), Sept. 21-24, Jaipur, India

ALICE,(2011).A.L.I.C.E. 2011. Artificial Intelligence Foundation [online].Available from:http:/ /alicepandorabots. com/ [Accessed 05/25/2011].

Shawar, Bayan Abu, and Eric Atwell. 2007. "ChatBots: are they really useful?." LDV Forum. Vol. 22. No. 1.

Md. ShahriareSatu, Md. HasnatParvez, 2015. "Review of integrated applications with AIML based ChatBot", 1st International Conference on Computer & Information Engineering, Dept. of CSE, Rajshahi University of Engineering & Technology, Rajshahi, Bangladesh.

M. J. Pereira, and L. Coheur, 2013. "Just. Chat-a platform for processing information to be used in ChatBots", Master Thesis, Lisboa

W. Xi, J. Lind and E. Brill. 2004. Learning Effective Ranking Functions for Newsgroup Search. InProceedings of SIGIR 2004, pp.394-401.

R. Nishimura, Y. Watanabe and Y. Okada. 2005. A Question Answer System Based on Confirmed Knowledge Developed by Using Mails Posted to a Mailing List. In Proceedings of the IJCNLP 2005, pp.31-36.

Alexandru Dobrila, 2010. From Semantic Web Knowledge to a Functional Conversational Agent: A Practical Approach.

Miguel Grinberg. 2014. Flask Web Development: Developing Web Applications with Python (1st ed.). O'Reilly Media, Inc.

Radim Rehurek and Petr Sojka, 2010. ˇ Software Framework for Topic Modelling with Large Corpora, Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks (Valletta, Malta), ELRA, pp. 45–50 (English).

Duplessis, Dubuisson, Vincent Letard, Anne-Laure Ligozat, and Sophie Rosset. 2016. "Joker chatterbot re-wochat 2016-shared task ChatBot description report." In RE-WOCHAT: Workshop on Collecting and Generating Resources for ChatBots and Conversational Agents-Development and Evaluation Workshop Programme.

Does Curriculum Learning help Deep Learning for Natural Language Generation?

Sandhya Singh[1], Kevin Patel[3], Pushpak Bhattacharyya[3],
Krishnanjan Bhattacharjee[2], Hemant Darbari[2], Seema Verma[1]

[1]Banasthali Vidyapith, Rajasthan, [2]C-DAC, Pune, [3]CFILT, IIT Bombay
sandhya.singh@gmail.com, {kevin.patel,pb}@cse.iitb.ac.in,
{krishnanjanb,darbari}@cdac.in, seemaverma3@yahoo.com

Abstract

Natural Language Generation (NLG) is a challenging problem in the field of Artificial Intelligence. The difficulty stems from the natural language's flexibility to convey the same message in different ways. Psycholinguists have always believed that learning simpler sentences early can lead to complex sentence creation using the same knowledge.This is also the intuition behind curriculum learning. Thus, in this paper, we investigate the use of curriculum learning for natural language generation. We show that curriculum learning is a promising training methodology for deep learning systems for NLG. We show this by reporting improvements obtained using i) a particular curriculum strategy and ii) augmenting data using curriculum logic. We use TGen, a deep learning based NLG system, for experimentation. We use 5 metrics for NLG evaluation and 8 metrics for readability evaluation. Our quantitative and qualitative evaluation shows that the system trained using curriculum methodology produces better quality text as compared to the system trained on normal data.

1 Introduction

Natural language generation involves creating a natural language(NL) sentence from a non-linguistic input, which can be a structured meaning representation (MR), statistical data or a parse tree structure. An example of MR to natural language is *[name(Barbeque Nation), eatType(restaurant)]* → *Barbeque Nation is a restaurant*. Ever since the pioneering work by (De Smedt et al., 1996; Reiter and Dale, 2000), this has been an active area of research (Gatt and Reiter, 2009; Lampouras and Androutsopoulos, 2013; Kondadadi et al., 2013; Wen et al., 2015b,a).

A major hurdle in applying traditional machine learning techniques for NLG is the possibility of having more than one correct sentences for a given meaning representation. In machine learning terminology, this implies two things : First, there can be more than one correct label for a given input. Second, the set of possible labels is infinite. Thus, computing the loss function, which is a standard component in many machine learning algorithms, is difficult (Lampouras and Androutsopoulos, 2013). Therefore, most approaches default to template based learning or guided learning mechanisms. The sentences generated using such approaches feel artificial, as there is not much variation in the generated sentences (Langkilde and Knight, 1998; Deemter et al., 2005; Manurung et al., 2008).

Deep learning has made tremendous strides in different areas of Natural Language Processing. Much of its success can be attributed to two factors: its ability to lend itself to representation learning and the emergence of set of techniques which make effective training of deep neural networks possible. Deep learning has been successfully applied in several NLP tasks : Part of Speech Tagging (Collobert and Weston, 2008), Sentence Classification (Kim, 2014), Sentiment Analysis (Liu et al., 2015), Sarcasm Detection (Joshi et al., 2016). Recently, Wen et al. (2015b); Dušek and Jurcicek (2016) *etc.* have investigated the use of deep neural networks to generate natural language.

Humans learn better through an organized

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 97–103

step-wise manner, exploiting already learned concepts while learning new difficult concepts. Tailoring training data in such a manner to assist a machine learning system is known as curriculum learning (Bengio et al., 2009; Fan et al., 2017).

We intuit that an NLG system may learn to form complex sentences by leveraging knowledge of forming simple sentences. Thus, in this paper, we raise the following question:

Does curriculum learning help improve the output quality and performance of deep learning based Natural Language Generation?

We investigate this question using TGen, a sequence to sequence based natural language generator (Dušek and Jurcicek, 2016). We performed a quantitative evaluation of the generated text. We also qualitatively evaluate coverage of MRs and ambiguity factor of the generated text. Our preliminary evaluations provide the following evidence for a positive reply to the above question:

- System trained using a length based curriculum strategy performs better than system trained using randomly shuffled data.
- System trained using curriculum-augmented data performs better than system trained on original data.

The rest of the paper is organized as follows: Section 2 describes related work to the problem. Section 3 provides the experimental setup of our evaluation. Section 4 discusses our quantitative and qualitative analysis followed by conclusion and future work.

2 Related work

2.1 NLG and Deep Learning

Chang et al. (2015) experimented with deep neural networks for sentence generation as well as other related features for generation. Wen et al. (2015a) used a joint recurrent and convolutional neural network for dialogue generation. Lampouras and Vlachos (2016) developed Locally Optimal Learning to Search (LOLS) framework, which used imitation learning to generate sentences. Wen et al. (2015b) proposed a semantically conditioned Long Short-term Memory(LSTM) for language generation. It was trained to learn from unaligned data.

Finally, Dušek and Jurcicek (2016) proposed TGen, a sequence to sequence based encoder-decoder architecture along with beam search and a reranker to generate natural language sentence from meaning representation. The architecture combines sentence planning and surface realization stages of generation and produces strings using LSTM based sequence generator.

These developments have led to a state where current NLG systems are able to generate more natural and varied output in comparison to earlier rule and template based generation. However, extremely large output space still leaves a lot scope for improvement.

2.2 Curriculum Learning, Deep Learning and NLP

The training criteria in most deep neural networks is non-convex. This adds two extra challenges to the problem of learning: the quality of the local minima obtained, and the speed of convergence towards that minima. Bengio et al. (2009) showed that curriculum learning addresses both these challenges positively. They demonstrated the effectiveness of curriculum learning for language modeling, among other tasks. This effectiveness was soon exploited by others. Shi et al. (2015) experimented with RNN language model for *within-domain adaptation* and *limited data within domain adaptation* using curriculum learning with improved outcomes. Cirik et al. (2016) studied the performance of curriculum learning on long-short term memory networks for sentiment analysis task. Similarly, Sachan and Xing (2016) showed that data ordering of simple hand crafted questions improved performance of question answering using deep neural networks. Liu et al. (2018) also experimented with curriculum learning approach for natural answer generation. This motivated us to explore NLG using curriculum learning.

Meaning Representation	name[Alimentum], area[city centre], familyFriendly[no]
Natural Language	Alimentum is not a family-friendly arena and is located in the city centre.

Table 1: Sample of input MR-NL utterance pair used for training

3 Experimental Setup

3.1 Data

3.1.1 Original Data

We used E2E-challenge dataset (Novikova et al., 2017). It is from the restaurant domain with around 42K sentences in the form of dialogue act-based meaning representations(MRs) coupled with its natural language utterances. The natural language text of the MRs from the dataset show open vocabulary with complex sentence structures and varied discourse patterns. Thus we conclude that this dataset is really good representative of the real world NLG problem. A sample MR and its NL utterance is shown in Table 1.

3.1.2 Creating curriculum-augmented data

Meaning Representation (MR)	Natural Language (NL)
name[Fitzbillies]	Fitzbillies is a restaurant .
name[Fitzbillies], eatType[coffeee shop]	Fitzbillies is a coffee shop .
name[Fitzbillies], food[French]	Fitzbillies serves french food .
name[Fitzbillies], area[riverside]	Fitzbillies is located in riverside .
name[Fitzbillies], eatType[coffeee shop], food[French]	Fitzbillies is a coffee shop serving french cuisine .
name[Fitzbillies], eatType[coffeee shop], area[riverside]	In the riverside area is a coffee shop named Fitzbillies .
name[Fitzbillies], food[French], area[riverside]	Fitzbillies serves french food in riverside area .
name[Fitzbillies], eatType[coffeee shop], food[French],area[riverside]	Fitzbillies is a coffee shop serving french food in riverside area.

Table 2: Sample of Curriculum data created from training MR: *"name[Fitzbillies], eatType[coffeee shop], food[French], area[riverside]"*

We create the curriculum-augmented data as follows. Let the original training set be $S_{original}$. Let the TGen model trained on randomly shuffled $S_{original}$ be $TGen_{original}$. For each MR of $S_{original}$, a set S_{comb} of all possible MR combinations with name field as constant factor is created. From this set S_{comb}, a subset S_{uniq_comb} of unique MR combinations is created. This fills the MR field of the dataset. To create the corresponding NL utterances, we first pass S_{uniq_comb} to $TGen_{original}$. The automatically generated utterances are then manually verified (for missing phrases corresponding to MR tag) and added to S_{uniq_comb}. This fills the NL field of the dataset.

We then remove duplicates from the combined set $S_{original} + S_{uniq\_comb}$ to create our training set $S_{augmented}$. The size of this $S_{augmented}$ dataset is 75K sentences. A sample of curriculum data is shown in Table 2.

3.2 Ordering Strategy

As discussed earlier, a major part of effectiveness of curriculum learning comes from the way data is ordered. We want the system to first learn to form simple sentences, and then move on to learning to form complex sentences. We define simplicity of a sentence in terms of the length of MR+NL. Thus in our experiments, we use the following ordering strategies:

- **Shuffled**: In this strategy, the data is randomly shuffled. This strategy is a baseline for comparison.
- **Curriculum**: In this strategy, the data is sorted based on the length of MR + NL.

3.3 Training

The training data was preprocessed to delexicalize the *name* and *near* MR slots to reduce the data sparsity, as recommended by Dušek and Jurcicek (2016). A sequence representation of each MR sequence is then created by joining the triplet information containing type, slot name and value for each MR slot and converted to a vector representation. We chose the string mode of TGen where it combines sentence planning and surface realization stages of NLG architecture(Konstas and Lapata, 2013). The TGen generator model was trained using a LSTM based seq2seq encoder-decoder architecture with 128 hidden units, embedding size 50, learning rate as 0.001 and batch size 20 along with Adam optimizer (Kingma and Ba, 2014). A reranker with beam size 10 is used for generation(Bahdanau et al., 2014).

We trained four different models as follows:

- **Sys1**: TGen trained on original data ($S_{original}$) with shuffled ordering strategy.
- **Sys2**: TGen trained on original data ($S_{original}$) with curriculum ordering strategy.
- **Sys3**: TGen trained on curriculum-augmented data ($S_{augmented}$) with shuffled

ordering strategy.
- **Sys4**: TGen trained on curriculum-augmented data ($S_{augmented}$) with curriculum ordering strategy.

3.4 Evaluation Metrics

We evaluated the quality (adequacy and fluency) of generated text using automatic evaluation metrics which measure the word-overlap with respect to the reference sentences - BLEU(Papineni et al., 2002), NIST(Doddington, 2002), METEOR (Banerjee and Lavie, 2005), ROUGE (Lin, 2004) and CIDEr (Vedantam et al., 2015) scores (as per the E2ENLG challenge). These metrics capture the degree of intended content that is transferred via the generated text.

We also evaluated the readability of the generated text using the automatic evaluation metrics which are:

1. The Flesch Reading Ease formula (FRE) (Flesch, 1948) that calculates the reading level of the content. It is scored from 1-100 with 100 being the easiest to comprehend and 1 being the hardest and confusing. A score of 60-70 is preferred for standard content.
2. The Flesch-Kincaid Grade Level (FKG) (Kincaid et al., 1975) score captures the level of content in the form of grade from 0-12 as per the standard accepted globally.
3. SMOG Index (SMOG) (Mc Laughlin, 1969) suggests the years of education needed to understand the piece of writing. The scores range from 5-18.
4. Gunning FOG Formula (GFOG) (Gunning, 1969) estimates the years of formal education needed to understand the text on the first reading. It ranges from 6-18.
5. Automated Readability Index (ARI) (Kincaid et al., 1975) is devised to gauge the understandability of a text. The scores grade the level needed to comprehend the text and varies from 1-12.
6. The Coleman-Liau Index (CLI) (Coleman and Liau, 1975) estimates the years of formal education required to understand the text on first reading with scores ranging from 1-12.
7. Linsear Write Formula (LWF) calculates the readability based on sentence length and no. of words with more syllables.
8. Dale-Chall Readability Score (DCRS) (Chall and Dale, 1995) measures the comprehension difficulty while reading the text.

One can infer from above that the major factors affecting these metrics are complexity of the vocabulary and average sentence length.

4 Results and Analysis

4.1 Quantitative Analysis

Table 3 shows the adequacy and fluency scores of the generated text. One can observe that, with respect to a particular dataset, the system trained using curriculum as ordering strategy is performing better than the system trained using shuffled ordering strategy **(Sys2 > Sys1 and Sys4 > Sys3)**. One can also observe that, with respect to a particular ordering strategy, the system trained using curriculum-augmented data is performing better than the system trained on original data **(Sys3 > Sys1 and Sys4 > Sys2)**. Overall, the system trained on curriculum-augmented data with curriculum as ordering strategy is performing the best.

Dataset	Original		Curr-aug	
Ordering	Shuf	Curr	Shuf	Curr
	Sys1	**Sys2**	**Sys3**	**Sys4**
BLEU	0.60	0.61	0.61	**0.64**
NIST	7.90	7.84	8.07	**8.39**
METEOR	0.41	0.42	0.41	**0.45**
ROUGE_L	0.66	0.67	0.66	**0.69**
CIDEr	1.95	2.00	2.00	**2.23**

Table 3: Results of adequacy and fluency evaluation. {original, curr-aug} indicates whether the training data was original or augmented. {Shuf, Curr} indicates the ordering strategy. Each value is the average of three runs. Each jump is statistically significant(p-values < 0.01)†

Next, we proceeded to measure the gains obtained using curriculum techniques. In this regard, we perform the remaining comparisons between the system without any curriculum learning component **(Sys1)** against the system with both curriculum learning components **(Sys4)**.

We observed that **Sys4's** percentage of MR tag coverage in the generated text is marginally higher (by 0.5953 percent points) than **Sys1's** coverage. Another observation

Metric	Sys1	Sys4	Ref
FRE	**71.80**	74.78	67.15
FKG	**6.43**	6.24	7.73
SMOG	6.9	**18.7**	11.6
GFOG	16.27	**16.45**	17.90
ARI	6.30	**6.44**	8.55
CLI	**8.00**	7.57	9.41
LWF	5.24	**5.77**	6.51
DCRS	7.85	7.85	8.23

Table 4: Results of readability evaluation. Except FRE, the higher the score the better

was that the average sentence length of text generated using **Sys4** was greater the average sentence length of text generated using **Sys1** by 2.36 percent points.

This shows that quantitatively, curriculum learning techniques can be helpful for deep learning based NLG.

Table 4 shows the readability evaluation of the generated text. The scores indicate that **Sys4** is generating relatively complex sentences as compared to **Sys1**.

4.2 Qualitative Analysis

Now we highlight some of the aspects where curriculum learning helped text generation.

MR	name[The Waterman], food[Fast food], priceRange[moderate], customer rating[3 out of 5], area[riverside], kidsFriendly[yes]
Sys1 Output	The Waterman is a kid friendly fast food restaurant with a moderate price range. It has a customer rating of 3 out of 5.
Sys4 Output	The Waterman is a kid friendly fast food restaurant in the riverside area with a moderate price range and a customer rating of 3 out of 5.

Table 5: Example showing how **Sys4** does handle the tag *area* which is dropped by **Sys1**

Consider the example in table 5. Here, it is evident that **Sys1** did not cover the *area[riverside]* term of the MR. Whereas, **Sys4** was able to accommodate it correctly.

Now, consider the example in table 6. Training data for both models included the Training MR and Training Reference, which had *food[French]* and *'is a French Pub'*. Now, the test MR has *food[English]*. One may observe that the **Sys1** generated a phrase similar to the reference, *i.e. 'is a English Pub'*. Note that this is a slightly ambiguous phrase, and can mean *a pub serving English food* or *a pub managed by English people*. Whereas, **Sys4** is able to generate a relatively unambiguous *'is a pub providing English food'* phrase. **Sys1** also has an ambiguous anaphora *'It'* (could mean both *The Plough* or *Cafe Rouge*), which is not the case with **Sys4**. Thus **Sys4** is generating better text here.

Training MR	name[The Plough], eatType[pub], food[French], priceRange[moderate], kidsFriendly[no], near[Cafe Rouge]
Training NL	The Plough is a French pub, which is not kid friendly. The price range is moderate and is located near caffe Rouge.
Test MR	name[The Plough], eatType[pub], food[English], priceRange[more than £30], children-friendly[yes], near[Cafe Rouge]
Sys1 Output of Test MR	The Plough is an english pub near Cafe Rouge. It is child friendly and has a price range of more than £30.
Sys4 Output of Test MR	The Plough is a pub providing english food in the high price range. It is located near Cafe Rouge and is children friendly.

Table 6: Example demonstrating how ***Sys4*** generates relatively unambiguous text

Using these examples, we qualitatively argue that curriculum system is better at NLG.

5 Conclusion and Future Work

In this paper, we proposed using curriculum learning as a training methodology for deep learning based natural language generation systems. We argued that both curriculum ordering strategy and curriculum augmented data could help learning natural language generation. Our quantitative evaluation showed that text generated using a system trained with either curriculum ordering strategy or curriculum-augmented data or both was better in terms of both adequacy and fluency, as well as readability when compared to a system trained on randomly shuffled original data. This was established via a set of different evaluation metrics. Our qualitative evaluation indicates that using curriculum led to better coverage and less ambiguity. Thus we conclude that curriculum learning based training methodology is indeed a promising methodology for deep learning based NLG systems. In the future, we will investigate MR based and vocabulary based approaches for designing curriculum strategies and data-augmentation for natural language generation.

[†]We used Welch unpaired t test for significance testing.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

Jeanne Sternlicht Chall and Edgar Dale. 1995. *Readability revisited: The new Dale-Chall readability formula*. Brookline Books.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daume III, and John Langford. 2015. Learning to search better than your teacher.

Volkan Cirik, Eduard Hovy, and Louis-Philippe Morency. 2016. Visualizing and understanding curriculum learning for long short-term memory networks. *arXiv preprint arXiv:1611.06204*.

Meri Coleman and Ta Lin Liau. 1975. A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60(2):283.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML)*, volume 307.0, pages 160–167. ACM.

Koenraad De Smedt, Helmut Horacek, and Michael Zock. 1996. Architectures for natural language generation: Problems and perspectives. In *Trends in Natural Language Generation An Artificial Intelligence Perspective*, pages 17–46. Springer.

Kees Van Deemter, Mariët Theune, and Emiel Krahmer. 2005. Real versus template-based natural language generation: A false opposition? *Computational Linguistics*, 31(1):15–24.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145. Morgan Kaufmann Publishers Inc.

Ondřej Dušek and Filip Jurcicek. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 45–51, Berlin, Germany. Association for Computational Linguistics.

Yang Fan, Fei Tian, Tao Qin, Jiang Bian, and Tie-Yan Liu. 2017. Learning what data to learn. *arXiv preprint arXiv:1702.08635*.

Rudolph Flesch. 1948. A new readability yardstick. *Journal of applied psychology*, 32(3):221.

Albert Gatt and Ehud Reiter. 2009. Simplenlg: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*, pages 90–93. Association for Computational Linguistics.

Robert Gunning. 1969. The fog index after twenty years. *Journal of Business Communication*, 6(2):3–13.

Aditya Joshi, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya, and Mark Carman. 2016. Are word embedding-based features useful for sarcasm detection? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1006–1011. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Technical Training Command Millington TN Research Branch.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ravi Kondadadi, Blake Howald, and Frank Schilder. 2013. A statistical nlg framework for aggregated planning and realization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1406–1415.

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *Journal of Artificial Intelligence Research*, 48:305–346.

Gerasimos Lampouras and Ion Androutsopoulos. 2013. Using integer linear programming for content selection, lexicalization, and aggregation to produce compact texts from owl ontologies. In

Proceedings of the 14th European Workshop on Natural Language Generation, pages 51–60.

Gerasimos Lampouras and Andreas Vlachos. 2016. Imitation learning for language generation from unaligned data. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1101–1112.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 704–710. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Cao Liu, Shizhu He, Kang Liu, and Jun Zhao. 2018. Curriculum learning for natural answer generation. In *IJCAI*, pages 4223–4229.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1433–1443. Association for Computational Linguistics.

Ruli Manurung, Graeme Ritchie, Helen Pain, Annalu Waller, Dave O'Mara, and Rolf Black. 2008. The construction of a pun generator for language skills development. *Applied Artificial Intelligence*, 22(9):841–869.

G Harry Mc Laughlin. 1969. Smog grading-a new readability formula. *Journal of reading*, 12(8):639–646.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. *arXiv preprint arXiv:1706.09254*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge university press.

Mrinmaya Sachan and Eric Xing. 2016. Easy questions first? a case study on curriculum learning for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 453–463.

Yangyang Shi, Martha Larson, and Catholijn M Jonker. 2015. Recurrent neural network language model adaptation with curriculum learning. *Computer Speech & Language*, 33(1):136–154.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Tsung-Hsien Wen, Milica Gasic, Dongho Kim, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015a. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. *arXiv preprint arXiv:1508.01755*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015b. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

WupLeBleu: The WordNet-Based Evaluation Metric for Machine Translation

Debajyoty Banik, Asif Ekbal, Pushpak Bhattacharyya
Department of Computer Science and Engineering
Indian Institute of Technology Patna, India
{debajyoty.pcs13,asif,pb}@iitp.ac.in

Abstract

In this paper, we propose a machine translation (MT) evaluation metric based on paraphrase matching fuzzy logic and the n-gram feature. Paraphrase matching generally calculates the relatedness between two strings by considering the depth, content, and structure in WordNet taxonomy. Various metrics based on stem match exist for MT evaluation. Since a sentence can be represented in different forms using synonyms and morphological structures, stem match is found inadequate to evaluate the MT output. Our proposed WupLeBleu evaluation metric can handle this challenge. Empirical evaluation on the benchmark datasets show that our proposed metric significantly improves the correlations with respect to the human judgment.

1 Introduction

The usage of automatic evaluation metric aims at evaluating the output quality of machine translation (MT) systems quickly. This is less expensive in comparison to the evaluation carried out by the trained experts. A few techniques were proposed for automatic evaluation. Especially, BLEU metric (Papineni et al., 2002) was widely used to automatically evaluate the quality of machine translation output. BLEU is an n-gram based method. However, weakness of BLEU was addressed in recent years (Ananthakrishnan et al., 2007). Many other automatic MT evaluation metrics like LeBleu (Virpioja and Grönroos, 2015), METEOR (Lavie and Denkowski, 2009), NIST (Doddington, 2002) etc. were proposed to overcome the issues of BLEU. To a huge degree, proposed WupLeBleu works on the principle of the fuzzy matching logic of the n-gram words along with the Wu-Palmer (WUP) similarity (Wu and Palmer, 1994) using WordNet. WUP similarity computes semantic relatedness of word senses using the edge counting method (Wu and Palmer, 1994).
We present an analysis of WupLeBleu with various language pairs: Chinese-English, Turkish-English, Czech-English, Russian-English, Finnish-English, and German-English.

1.1 Related Works

The most popular automatic MT evaluation metric is BLEU that computes n-gram matchings of the candidate (C) with reference (R) translation. It computes the overall precision of n-grams by using geometric average along with the brevity penalty. But, there are many issues with the automatic evaluation of BLEU metric, as it solely focuses on the n-gram matchings. Researchers proposed NIST (Lin and Hovy, 2003) to calculate the score based on the information gain from each n-gram. NIST evaluation assigns more score to the n-gram which is more informative.METEOR (Lavie and Denkowski, 2009) is based on explicit word-to-word matchings using the stem, and synonym modules. RIBES was proposed (Neubig et al., 2012) with the primary focus on the word order of a sentence and by considering the brevity penalty for calculating the final score with the help of Kendall's correlation. Very recently, researchers introduced LeBleu (Virpioja and Grönroos, 2015) that considers

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 104–108

fuzzy based matching and computes the similarity score based on Levenshtein distance. LeBleu uses arithmetic averaging for calculating the overall precision of score. LeBleu cannot handle paraphrase or synonym. This is regarded as one of the major drawbacks. The proposed WupLeBleu is designed in such a way so that it can properly handle all of the challenges like synonym matching, fuzzy matching and morphological differences altogether.

2 Issues in Existing Machine Translation Evaluation Metrics

Despite the fact that the BLEU is widely used metric for MT evaluation, it experiences a few shortcomings which we particularly intend to address in our proposed metric.

1. BLEU, a precision based metric that matches word n-grams of MT-translation output with multiple reference translations simultaneously. Lack of attention to recall within BLEU is a great shortcoming. The "Brevity Penalty" in the BLEU metric does not satisfactorily compensate for the absence of recall.

2. The n-gram matching focuses exact word matches and all the matched words weigh equally in BLEU. The geometric average of n-gram scores produces a result of zero if the individual n-gram scores are zero.

3. The correlation between BLEU score and human evaluation is very poor (Ananthakrishnan et al., 2007).

For example, let us consider the candidate and reference translations as stated below:
C: *H*e who fears as a result of conquered is a sound of defeat.
R: *H*e who fears being conquered is sure of defeat.
Here, R and C refer to reference and candidate translation of phrase-based statistical machine translation (PBSMT) system, respectively. The computed BLEU score will be zero for C, because of the absence of the four-gram matchings in C1 when checked against the reference translations.
C: *T*he 7^{th} era are as yet battling for their rights.
R: *T*he seventh generation is still fighting for their rights.
For example, both BLEU and LeBleu fail as the n-gram matchings are absent. METEOR, which considers only the precision of uni-gram matchings calculates the score based on explicit word-to-word matching. The default METEOR parameters prefer longer translations than the other metrics. Since precision and recall are computed for uni-gram matching, the high α values contribute more weight to uni-gram recall than precision. This puts METEOR in disadvantage position when being evaluated by the other metrics. The primary objective of our proposed WupLeBleu metric is to overcome the problems as mentioned above. Consider the following example. Here, both candidate and reference translations convey the same meaning, but with different vocabularies.

C: हर स्थान शांति छा गया।
ETL: *h*ar sthaan shaanti chha gaya.
ET: *E*very place has peace.
R: हर जगह सन्नाटा छा गया।
ETL: *h*ar jagah sannaata chha gaya.
ET: *S*ilence everywhere.
Here, ETL, ET are the English transliteration and English translation, respectively. But, the computed BLEU score would be zero, as exact n-gram matchings are absent. Also, LeBleu partially solves this problem by using the fuzzy matching technique. But, WupLeBleu metric has the power of solving a fuzzy n-gram matching technique along with the WUP similarity.

3 Methodology

WupLeBleu calculates the score based on the precision of n-gram matching with fuzzy logic along with the WUP similarity score. The WUP similarity score uses WordNet to improve the correlation of automatic evaluation metric with human evaluation. This score provides the detailed idea of candidate words with respect to reference words in terms of synonyms and lemmas. The WUP similarity method (Wu and Palmer, 1994) generally calculates the relatedness between the two words by considering the depth, content and structure of two strings in WordNet taxonomies. The similarity measure is computed

based on the ratio of the information content of the least common subsumer of the candidate and the reference string. LCH (Leacock and Chodorow, 1998), the WUP similarity (Wu and Palmer, 1994) and the path length are three similarity measures that are considered based on the path length (Pedersen et al., 2004) between C and R sentences. LCH method calculates the minimum path between the source and the target string, and then scales the minimum path by the maximum path length found in the hierarchy in which they occur. The WUP similarity score is calculated as the sum of the depth of LCS (Least Common Subsumer) between the words from C and R sentences. The path score is equal to the inverse of the shortest path between two strings (Pedersen et al., 2004). The final WUP similarity score is calculated based on the above three measures.

$$WUP\,similarity = 2 * \frac{depth(lcs)}{(depth(s1) + depth(s2))}$$

If the WUP similarity score is more than the predefined threshold parameter δ, then both the words are considered to be nearly similar and their matching n-gram precision is taken into account while calculating the overall n-gram precision, else ignored. Fuzzy matching works on the fact, that the n-gram matching is said to be a fuzzy match if the similarity score is more than the threshold parameter ∂. The fuzzy based similarity score is calculated as one minus letter edit distance. The letter edit distance (levenshtein distance) is a measure of the similarity between two strings (Heeringa, 2004). The distance ($lev_{a,b}(i,j)$) is calculated by the required number of insertions, deletions, or substitutions, to transform a source into target string.

$$lev_{a,b}(i,j) = \begin{cases} max(i,j) & if\ min(i,j) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{a_i \neq b_j} \end{cases} & otherwise \end{cases}$$

The brevity penalty (BP) considers the total number of characters rather than the words present in reference and candidate translations. Overall precision is calculated by combining the individual n-gram precisions through arithmetic averaging.

$$BP = \begin{cases} 1 & c > r \\ e^{(1-(r/c)} & c \leqslant r \end{cases}$$

Here, the variables r and c refer to the total number of characters in the reference and candidate translations respectively.

4 Experiments

We conduct the experiments on WMT 14 dataset (Machacek and Bojar, 2014) and HinEnCorp (Bojar et al., 2014) for five different language pairs. At first, the WUP similarity scores are calculated among the words of the aligned sentence. If this WUP similarity score is more than the tuned threshold parameter δ, then two words are considered as a matching. After fine tuning we found δ value as 0.80. If there is a matching of more than one single word pair, then the word pair with greater WUP similarity value will be chosen as matching. All synonyms, morphological structure, and other representation of the words are covered by using this step. It then calculates the similarity score which uses fuzzy logic. Basically, it calculates the Levenshtein distance between the two strings. TH final score is then computed by calculating the arithmetic average of the individual n-gram matchings multiplied by brevity penalty (BP).

5 Performance in WMT 14 Dataset

We also evaluate our proposed algorithm using WMT 2014 dataset [1]. The highest correlation with human judgment is found for Hindi to English (hi-en) and French to English (fr-en) language pairs. After calculating the correlation with human judgment on average, we found the score as 0.951. This shows that our proposed model stands out on top, considering the average score. In most of the cases the proposed metric achieves better correlation than the standard metrics, shown in Table 1.
Main challenge in WUP similarity approach is to tune δ. If this value is too small then synonymous words may not be considered as similar words. For large value of δ may cover distance words as similar one. For example, this

[1] http://www.statmt.org/wmt14/

Table 1: Comparison: Correlation with different metrics in WMT 14 Dataset

Metric	Pearson Correlation					
	de-en	ru-en	cs-en	fr-en	hi-en	Average
WupLeBleu	**0.931**	**0.882**	**0.985**	**0.973**	**0.984**	**0.951**
LeBleu	0.892	0.896	0.912	0.971	0.969	0.928
LAYRED	0.893	0.843	0.940	0.973	0.976	0.925
BLEU	0.831	0.774	0.908	0.952	0.956	0.884
NIST	0.810	0.785	0.983	0.955	0.783	0.863
METEOR	0.926	0.792	0.980	0.975	0.457	0.826
TER	0.774	0.796	0.977	0.952	0.618	0.823

metric may identify ”foot-ball” and ”basket-ball” as similar words which is not true.
We have done significance tests, and observe that results are significant with 95% confidence level (with p=0.1 which is < 0.05).

6 Evaluation with other Datasets

We evaluate the WupLeBleu for English to Hindi (en-hi) translation. Due to the unavailability of en-hi language in WMT dataset we study the proposed evaluation score by using miscellaneous domain data sets from the HinEnCorpora. We choose three systems: Moses's default configuration for SMT system[2], Google[3] and Bing[4] translator) for the correctness checking of our proposed metrics. We take 271877 and 1001 sentence pairs for training and tuning of SMT, respectively. For evaluation we use 1002 sentence pairs. After detailed analysis (with 1002 sentence pair), we achieve better Pearson correlation for the proposed WupLeBleu. The Pearson correlations are BLEU: 0.9103, METEOR: 0.9137, Lebleu: 0.9278 and WupLeBleu: 0.9434.

We also manually evaluate the F-beta scores (Figure 1) for different automatic evaluation metrics and compare their ratio (Figure 2) to estimate how close these are to human evaluation. It is clearly understood from Figure 2) that LeBleu and proposed WupLeBleu's evaluation preferences are closer to manual judgment. WUP similarity makes WupLeBleu better. We have added details of manual evaluation in the additional sheet.

[2]http://www.statmt.org/moses/manual/manual.pdf
[3]https://translate.google.com/
[4]https://www.bing.com/translator

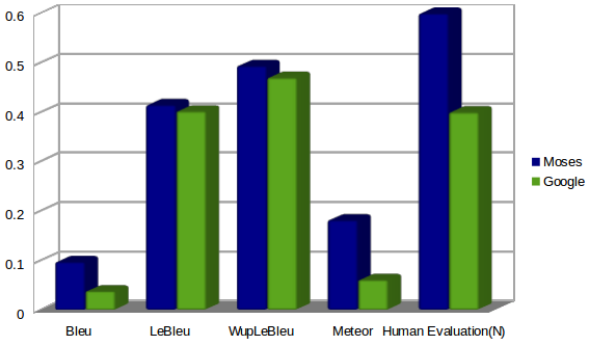


Figure 1: Ranking of correctness (hi-en)

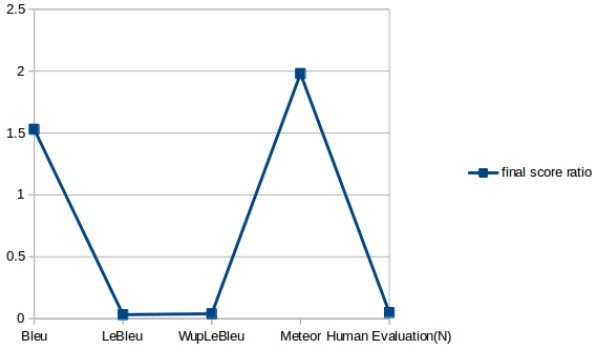


Figure 2: Score ratio between dataset-1 and dataset-2

7 Conclusion

In this paper we have proposed an automatic MT evaluation metric, WupLeBleu. Based on a large study and several experiments, we can conclude that fuzzy logic based n-gram matching with the WUP similarity method can perform more accurate MT evaluation than the existing metrics. We believe that our proposed approach that uses WUP similarity and fuzzy logic has a higher similarity to human evaluation. In future we will also evaluate the WupLeBleu metric on the other language pairs.

References

R Ananthakrishnan, Pushpak Bhattacharyya, M Sasikumar, and Ritesh M Shah. 2007. Some issues in automatic evaluation of english-hindi mt: more blues for bleu. *ICON*.

Ondrej Bojar, Vojtech Diatka, Pavel Rychlỳ, Pavel Stranák, Vít Suchomel, Ales Tamchyna, and Daniel Zeman. 2014. Hindencorp-hindi-english and hindi-only corpus for machine translation. In *LREC*, pages 3550–3555.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145. Morgan Kaufmann Publishers Inc.

Wilbert Jan Heeringa. 2004. *Measuring dialect pronunciation differences using Levenshtein distance*. Ph.D. thesis, University Library Groningen][Host].

Alon Lavie and Michael J Denkowski. 2009. The meteor metric for automatic evaluation of machine translation. *Machine translation*, 23(2):105–115.

Claudia Leacock and Martin Chodorow. 1998. Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database*, 49(2):265–283.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 71–78. Association for Computational Linguistics.

Matous Machacek and Ondrej Bojar. 2014. Results of the wmt14 metrics shared task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 293–301.

Graham Neubig, Taro Watanabe, and Shinsuke Mori. 2012. Inducing a discriminative parser to optimize machine translation reordering. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 843–853. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet:: Similarity: measuring the relatedness of concepts. In *Demonstration papers at HLT-NAACL 2004*, pages 38–41. Association for Computational Linguistics.

Sami Virpioja and Stig-Arne Grönroos. 2015. Lebleu: N-gram-based translation evaluation score for morphologically complex languages. In *WMT@ EMNLP*, pages 411–416.

Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics.

"Is This A Joke?": A Large Humor Classification Dataset

Faraz Faruqi
Manipal Institute of Technology
Manipal
faruqi.faraz@gmail.com

Manish Shrivastava
IIIT Hyderabad
Hyderabad
m.shrivastava@iiit.ac.in

Abstract

Humor is an essential characteristic of language. It has been a topic of research in linguistics and philosophy from historical times. In computer science, computational humor, as a part of Natural Language Processing, is a growing area of research. Social Media is rapidly growing as a platform for communication but processing of social media, owing to its semantic perplexity, is still a challenge. These two facts lead us to present a novel dataset for humor classification which captures diversity in humor on web resources. The large size of this dataset is to meet the data requirements for modern machine learning algorithms. This paper also deals with creating a model for detecting and analyzing humor in social media text extracted from eclectic sources on the Internet.

1 Introduction

Humor is one of the most interesting aspects of human language and behavior. Humans have an innate sense of humor. They can understand, interpret and create humor almost effortlessly. But even though it is a part of our daily conversations, computationally detecting and analyzing humor remains a challenge. In recent years, the study of humor has also developed into an area of computational research under computational linguistics (Friedland and Allan, 2008; Mihalcea and Pulman, 2007; Kiddon and Brun, 2011).

Among all the theories of humor (Attardo, 2010), one of the most widely accepted is the *'Incongruity Theory'* (Morreall, 1986). It suggests that humor is due to the mixing of two disparate interpretation frames in one statement.It has recently been formalized as a necessary condition for humor and used as a basis for the Semantic Script-based Theory of Humor (SSTH) (Raskin, 2012).

Two fish in a tank. One turns to the other and says: "Do you know how to drive this?"

The incongruity theory can also be explained as a theory of comprehension. As the joke gradually evolves, two linear train of thoughts emerge, leading to the obvious and *latent* meanings respectively. As the joke nears its end, with a clever play of words, the latent meaning becomes the dominant one and ends up being the punch line of the joke. In the preceding example, the first sentence has two connotations. "*Fish* in a Tank" and "Fish in a *Tank*". By taking the first interpretation, reader assumes the tank in question to be a fish tank. But the statement - *"Do you know how to drive this?"*, suddenly converts the connotation to one where the 'tank' in question is an 'armored car'. This creates a sense of surprise and makes the sentence humorous. Jokes based on the incongruity theory use clever wordplay to elucidate humor in a sentence. They are also short in length, making them particularly suitable for an automatic learning setting. Such kind of humor is popular among the ubiquitous social media websites, and dedicated webpages. But it is essential to standardize this data before using it in an automatic setting.

2 Related work

Previous work in computational humor had focused mainly on the task of humor generation (Binsted and Ritchie, 1997; Stock and Strapparava, 2003), and there has been a relatively recent paradigm shift towards humor detection. Previous researchers (Purandare and Litman, 2006; Mihalcea et al., 2010) have used a set of linguistic features to detect them (polysemy, alliteration, antonyms and adult slang etc). Kiddon and Brun

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 109–114

(2011) recognized a subproblem (*Double Entendre Identification*) and constructed models to detect sexual euphemisms or wordplay in sentences. Similarly, Purandare and Litman (2006) analyzed humorous spoken conversations from a classic comedy television show - *"FRIENDS"*, by examining acoustic-prosodic and linguistic features. Taylor and Mazlack (2004a,b) considered a restricted set of all possible jokes that had wordplay as a component and examined the limited domain of *"Knock Knock"* jokes. Also, there have been other interesting researches, such as by Yang et al. (2015) where the authors developed models to extract humor 'anchors' from the sentences.

In the domain of humor-analysis on social media, Barbieri and Saggion (2014) developed automatic models to detect irony in sentences from Twitter; models developed by Davidov et al. (2010) did a semi-supervised recognition of sarcasm on Twitter and Amazon reviews. By taking the context of the tweet into account while classifying, Bamman and Smith (2015) obtained higher accuracies in detection of sarcasm in tweets.

There have not been many efforts to use deep learning methods in humor detection, owing to the subjectivity of the task and requirement of huge amount of data. de Oliveira and Rodrigo (2015) used RNN and CNN models to detect humor. But this work was in the limited domain of Yelp Reviews, and we have tried to extend this problem to the language used in social media.

3 Dataset

As rightly specified by de Oliveira and Rodrigo (2015), there is no large body of work on so-called *"computational humor"*. Work that exists is largely in the pure NLP domain and uses hand-written features and simplistic tree methods or SVMs (Mihalcea and Strapparava, 2006; Yang et al., 2015). This is because there is no such labeled corpus of funny texts available for a detailed semantic analysis. This problem can also be attributed to the subjectivity of assigning a binary outcome onto something as complex as humor (Bamman and Smith, 2015).

We also felt that a dataset was needed that contained balanced counts of positive and negative samples of humor. The previous dataset created by Mihalcea and Strapparava (2005) contains 16000 one-liners, obtained through bootstrapping method. Their work on this dataset in (Mihalcea and Strapparava, 2006) obtained the state-of-the-art accuracy in humor detection. de Oliveira and Rodrigo (2015), have extracted sentences from *Yelp Review* dataset. The 16000 one-liner dataset, although containing balanced samples, was too small to train a large network. The *Yelp Review* was also inapplicable in this case as it contained very long samples, extending for more than one sentence. Hence, the authors created this new dataset containing huge number of one-liner jokes, containing 400,000 sentences extracted from dedicated humor pages on various social media websites.

3.1 Our Dataset

This dataset contains 400,000 sentences extracted from social media and humor-dedicated websites. We used Reddit's PRAW API[1] Twitter's REST API[2] (Makice, 2009) to extract samples from various humor-dedicated pages from these sources. We also used Web Scraping (Munzert et al., 2014) method to extract a large amount of sentences from various dedicated websites like *jokeoftheday.com*, *wocka.com*, *short-funny.com*, *onelinefun.com*. Permission for scraping were taken from website owners and maintainers whenever necessary.

The negative samples were carefully chosen for testing the model on multiple settings. The hypothesis was that as the semantic similarity between the negative and positive samples increases, the accuracy of classification decreases. The sources for negative samples are:

1. News headlines from Reuters' News Agency website spanning a period of 5 years [3].
2. British National Corpus (Consortium et al., 2012)
3. Proverbs (Extracted from an online proverb collection)

4 Preprocessing and Balance

The sentences found on different sources were slightly different in terms of style and content. Thus, for the proper training of models, a standardization procedure was followed, which has summarized below.

The sentences containing any sort of code-mixing, image or hyperlink were removed.For

[1] https://praw.readthedocs.io/en/latest/
[2] https://developer.twitter.com/en/docs
[3] https://www.thomsonreuters.com/en.html

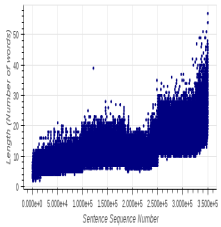
standardizing the usage of punctuation, we limited the number of punctuation marks to a max of three repetitions concurrently. CamelCase words were separated into distinct words (Friedl, 2002). Digits were separated from the alphabets using regular expressions. The non-humor samples were normalized similarly. In the Reuters dataset, highly repetitive phrases like *'Stock Market value'* and *Breaking News* were removed.

In order to cross-verify the integrity of the dataset, we randomly sampled 100 instances of humor from each source, and confirmed whether they were indeed humorous or not.

The following table contains statistical information about the Humor, Non-humor corpus and the 16000 sentences dataset (Mihalcea et al., 2010). The statistics are in terms of **Sentence Sequence number** (x-axis) vs **Number of words** (y-axis) of each sentence. It can be inferred from the data presented that the distribution of the previous dataset was much more balanced in terms of lengths of jokes. We did not maintain such a balance in our dataset as it would inhibit the model's capacity of capturing the kind of humor encountered in the real world and the social media. We intentionally included sentences of different lengths to aid diversity in the dataset.

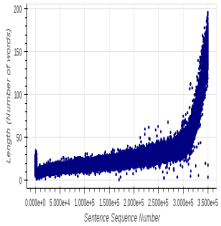
HUMOR-

Minimum Length: 3
Maximum Length: 250
Mean: 30
Median: 22.16
Mode: 16
Population Variance: 698.28



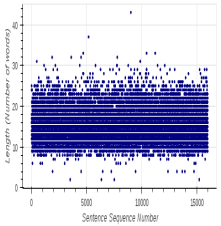
NON HUMOR-

Minimum Length: 2
Maximum Length: 257
Mean: 11
Median: 10.69
Mode: 9
Population Variance: 22.92



16000 One-liners dataset

Minimum Length: 2
Maximum Length: 43
Mean: 15
Median: 14.94
Mode: 14
Population Variance: 14.76



5 Experiments

Humor detection has been recognized as a binary text classification problem. But humor detection is a difficult task. Detecting incongruity in humor means that the model has to predict when a word is being used for multiple meanings (polysemy), or has to detect the change of focus in the sentence.

In order to classify the sentence based on the presence of incongruity, we try and detect the wordplay in the sentence by analyzing the relationships between the words. Word vector representations of(Mikolov et al., 2013a,b) the words were used to evaluate such relationships. In order to find the sentence-embedding from the words, two methods were used-

1. Unweighted averaging of the word vectors.
2. An RNN based language model (Cho et al., 2014).

In order to optimize the results, we performed experiments with both Word2Vec Skip Gram model (Mikolov et al., 2013b) and a count based *GloVe* Vector representation (Pennington et al., 2014) and the results have been shown below.

A minimum frequency of five has been used for creating these vectors. The window size for the embeddings was set to be 10.

5.1 Classification

Our sentence embedding is a Bag-of-Words (Zhang et al., 2010) vector averaging. We compare this with a neural language modeling based sentence embedding in the following sections.

5.1.1 Method of Averaging

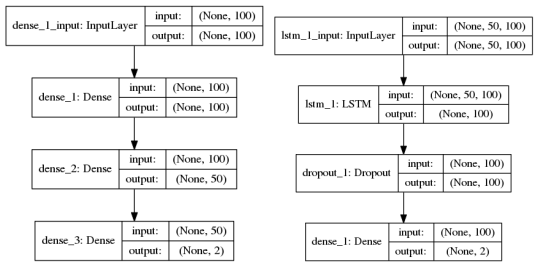
Let $\vec{x_i} \in \Re^k$ be a k-dimensional (k = 100) vector corresponding to the i-th word in the sentence. Then the sentence of length N can be represented as

$$\vec{X_i} = \frac{1}{N} \sum_{n=1}^{n=N} \vec{x_i} \quad (1)$$

The two classifiers used for the task are: Artificial Neural Networks (MLP) and Support Vector Machines (Tong and Koller, 2001). For the MLP classifier, as shown in Figure 1a, we used a 3 layered, densely connected network. The activation function used for the hidden layers was 'reLU' (Rectified Linear Unit) (Equation 2) (Dahl et al., 2013) (commonly known as reLU)

$$f(x) = max(x, 0) \quad (2)$$

and a softmax function at the last layer. In the SVM classifier, we used the *rbf* (Radial Basis Function) as the kernel (Hsu et al., 2003) and found the optimum parameters using an exhaustive grid search.



(a) MLP (b) LSTM

Figure 1: Network Architectures

5.1.2 Neural Language Model

The Recurrent Neural Network (RNN) (Cho et al., 2014; Bengio et al., 2013) is a natural generalization of feedforward neural networks to sequences (Sutskever et al., 2014). Given a sequence of inputs $(x^1, ..., x^T)$, an RNN encoder encodes it into a single vector c by iterating over the following equation:

$$h_t = f(x_t, h_{t-1})$$

We use an LSTM (Hochreiter and Schmidhuber, 1997) layer instead of RNN to capture long-term dependencies (Bengio et al., 1994).

The final encoded vector c results from the equation $c = q(\{h_1, ...h_{T_x}\})$, (Cho et al., 2014) where $q(\{h_1, ..., h_T\}) = h_T$ for the case of LSTMs , as presented by (Sutskever et al., 2014). This encoded vector is used as the sentence embedding for classification purposes. The model was created using APIs by Tensorflow (Abadi et al., 2015) and has been presented in Figure 1b.

The sentences were pre-padded up to the length of 50 steps. We used stochastic gradient descent (SGD) algorithm together with Adadelta (Zeiler, 2012) to train the model. In order to tackle the difficulty of different lengths of sentences, we also used a variable-length vector (Dynamic Length RNN) (Cho et al., 2014) and found better results.

5.2 Results

Technique	Reuters	BNC	Proverbs
MLP	99.46%	75.6%	81.4%
SVM	96.2%	71.5%	78.6%
LSTM	74.20%	52.2%	55.6%

Table 1: Comparison of results obtained using different datasets as negative samples.

The dataset was split into 80%-10%-10% (training - validation - testing) sets to train and test the model. Cross-validation was done through the dataset and the results were averaged. The final accuracy achieved was 99.46%.Empirically, it was observed that pre-trained vectors performed much better than the other freshly trained GloVe vectors and Word2Vec (Rehurek and Sojka, 2011) vectors. This can be attributed to the fact that they were trained on a much larger dataset, hence they better captured the substructures of the vector space.

Technique	Accuracy
Our Approach	95.8%
(Mihalcea et al., 2010)	**96.89%**
(Yang et al., 2015)	80.5%

Table 2: Comparison with other results from the literature on 16000 One-Liners Dataset.

As presented in table 1, LSTMs were not able to classify the sentences as accurately as MLP and SVM. We postulate that in order to perform well at humor classification, a much larger dataset is required for training the LSTM model. In Table 2, we report the efficacy of using different datasets as negative samples. Since News' headlines are semantically most dissimilar from humorous sentences, and the sentences from the BNC corpus are most similar, the accuracy is the highest in the former case, and the lowest in the latter.

We also tested our model on the 16000 One-Liners (Mihalcea and Strapparava, 2006) dataset, and got comparable results. It should be noted that since the humorous samples found in this dataset were extracted from a different source. This sort of cross-domain classification experiment proves that this approach can be generalized.

6 Conclusion and Future work

We have presented a methodology for detecting humor in social media text. A new dataset has been created and used to train machine learning models that can detect humor in English sentences. We are releasing this comprehensive dataset that has been standardized to be used in an NLP setting. We present our approach towards humor detection, along with the results achieved. Experimental results display the applicability of the model. Since LSTMs did not give acceptable results in classification, it would be interesting to use *'Attention'* (Bahdanau et al., 2014), and increase the dataset size to train such a model. In the future, we would also like to extend our work in computational humor to humor generation.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Salvatore Attardo. 2010. *Linguistic theories of humor*, volume 1. Walter de Gruyter.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

David Bamman and Noah A Smith. 2015. Contextualized sarcasm detection on twitter. In *ICWSM*, pages 574–577.

Francesco Barbieri and Horacio Saggion. 2014. Automatic detection of irony and humour in twitter. In *ICCC*, pages 155–162.

Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. 2013. Advances in optimizing recurrent networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8624–8628. IEEE.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Kim Binsted and Graeme Ritchie. 1997. Computational rules for generating punning riddles. *HUMOR-International Journal of Humor Research*, 10(1):25–76.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

BNC Consortium et al. 2012. The british national corpus, version 3 (bnc xml edition). 2007. *Distributed by Oxford University Computing Services on behalf of the BNC Consortium. URL: http://www. natcorp. ox. ac. uk (last accessed 25th May 2012)*.

George E Dahl, Tara N Sainath, and Geoffrey E Hinton. 2013. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8609–8613. IEEE.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the fourteenth conference on computational natural language learning*, pages 107–116. Association for Computational Linguistics.

Jeffrey EF Friedl. 2002. *Mastering regular expressions*. " O'Reilly Media, Inc.".

Lisa Friedland and James Allan. 2008. Joke retrieval: recognizing the same joke told differently. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 883–892. ACM.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. 2003. A practical guide to support vector classification.

Chloe Kiddon and Yuriy Brun. 2011. That's what she said: Double entendre identification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 89–94. Association for Computational Linguistics.

Kevin Makice. 2009. *Twitter API: Up and running: Learn how to build applications with the Twitter API*. " O'Reilly Media, Inc.".

Rada Mihalcea and Stephen Pulman. 2007. Characterizing humour: An exploration of features in humorous texts. *Computational Linguistics and Intelligent Text Processing*, pages 337–347.

Rada Mihalcea and Carlo Strapparava. 2005. Computational laughing: Automatic recognition of humorous one-liners. In *Proceedings of Cognitive Science Conference*, pages 1513–1518.

Rada Mihalcea and Carlo Strapparava. 2006. Learning to laugh (automatically): Computational models for humor recognition. *Computational Intelligence*, 22(2):126–142.

Rada Mihalcea, Carlo Strapparava, and Stephen Pulman. 2010. Computational models for incongruity detection in humour. *Computational linguistics and intelligent text processing*, pages 364–374.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

John Morreall. 1986. The philosophy of laughter and humor.

Simon Munzert, Christian Rubba, Peter Meißner, and Dominic Nyhuis. 2014. *Automated data collection with R: A practical guide to web scraping and text mining*. John Wiley & Sons.

Luke de Oliveira and Alfredo Láinez Rodrigo. 2015. Humor detection in yelp reviews.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Amruta Purandare and Diane Litman. 2006. Humor: Prosody analysis and automatic recognition for f* r* i* e* n* d* s. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 208–215. Association for Computational Linguistics.

Victor Raskin. 2012. *Semantic mechanisms of humor*, volume 24. Springer Science & Business Media.

R Rehurek and P Sojka. 2011. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2).

Oliviero Stock and Carlo Strapparava. 2003. Getting serious about the development of computational humor. In *IJCAI*, volume 3, pages 59–64.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Julia M Taylor and Lawrence J Mazlack. 2004a. Computationally recognizing wordplay in jokes. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 26.

Julia M Taylor and Lawrence J Mazlack. 2004b. Humorous wordplay recognition. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 4, pages 3306–3311. IEEE.

Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66.

Diyi Yang, Alon Lavie, Chris Dyer, and Eduard H Hovy. 2015. Humor recognition and humor anchor extraction. In *EMNLP*, pages 2367–2376.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Yin Zhang, Rong Jin, and Zhi-Hua Zhou. 2010. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52.

Fuzzy Evolutionary Self-Rule Generation and Text Summarization

Pradeepika Verma
Department of CSE
IIT (ISM) Dhanbad, India
pradeepikav.verma093@gmail.com

Hari Om
Department of CSE
IIT (ISM) Dhanbad, India
hariom4india@gmail.com

Abstract

Over the past decades, soft computing and statistical techniques has acquired serious attention for solving machine learning problems. A system consisting of human like capabilities can be developed using fuzzy logic. Moreover, evolutionary techniques are often fit for approximating solutions. With these potentials, we propose a supervised text summarization approach based on fuzzy rules and human-engineered features. A data-driven fuzzy rules generation system is modeled as a discrete optimization problem and then used to classify the sentences of document. According to these classifications, the relevant sentences are extracted for summary generation. The experimental results on DUC2006 dataset show the effectiveness of the proposed model.

1 Introduction

The automatic text summarization (ATS) systems are welcomed since the necessity to access large amount of textual data has grown. The main objective of these systems is to fetch significant information from the document while maintaining the user requirement (Mani, 2001) and challenge is to produce high quality summary (Binwahlan et al., 2010). According to Ferreira et al. (2013), an ATS system generates concise form of single or multiple documents. Although, a lot of work has previously been done in the field of extractive summarization, the challenges are still there.

In this paper, the proposed model focuses on extractive summary generation based on fuzzy logic and text features. Most of the existing extractive methods are based on finding the relevant sentences using text features such as sentence position and length (Erkan and Radev, 2004), existence of title words, frequent words, and proper nouns in the sentence (Nenkova et al., 2006; Edmundson, 1969). The scores of these features are typically used to assign the score to sentences for making decision on the relevancy of sentences. However, decision with these scores sometimes become fuzzy and low (Zha, 2002; Lin and Hovy, 2003; Murad and Martin, 2007). For example, suppose there are two sentences s_1 and s_2 evaluated on the basis of two features f_1 and f_2 with their weights 0.9 and 0.2, respectively. The feature's scores of s_1 are 0.1 and 0.9 and s_2 are 0.3 and 0.0. According to these scores, both s_1 and s_2 get 0.27 score and consists of equal priority for selection. But if we observe, the feature f_1, whose score is negligible in s_1, has much higher weight in comparison to f_2. Therefore, it would be appreciable if s_2 gets higher priority than s_1. This condition can be better handled with fuzzy logic which motivates us to explore it for summarization.

Few of research works till date have been devoted to fuzzy based summarization schemes. Binwahlan et al. (2010) proposed a fuzzy swarm based summarization method where every sentence is computed on the basis of their feature scores adjusted by particle swarm optimization generated weights. Their scores are then applied to fuzzy logic for better classification in important and unimportant sentences. Abbasi-ghalehtaki et al. (2016) also applied fuzzy logic in the same manner for summarization task where scores are adjusted by hybrid genetic and particle swarm optimization algorithms. In these schemes, much effort and time has been devoted to create fuzzy rules through human experts. Moreover, the processing with these huge set of rules is also time consuming. It motivates us to generate a data-driven optimal set of fuzzy rules. According

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 115–119

to Binwahlan et al. (2009), fuzzy logic and swarm intelligence could perform better in the field of text summarization. Therefore, the optimization power of particle swarm optimization (PSO) algorithm in discrete form has been utilized to generate fuzzy rules in this paper.

2 Problem formulation

In this section, we formally introduce our model as follows. Suppose $D = \{s_1, s_2, \ldots, s_n\}$ is a document which consists of n number of sentences where s_m denotes m^{th} sentence of the document. At first, sentence segmentation, and stop word removal steps are carried out. As a result, each sentence $s_m = \{w_1, w_2, \ldots, w_{|s_m|}\}$ is converted into a set of keywords. Thereafter, we score the sentences according to seven text features as given in Table 1.

Text feature	Description	Formulation
f_1	Title words (tw)	$\frac{\sum_{tw \in s_m} (Count(gram_N)}{\lvert tw \rvert . \lvert s_m \rvert}$
f_2	Proper noun (pn)	$\frac{\sum_{pn \in s_m} Count(gram_N)}{\lvert pn \rvert . \lvert s_m \rvert}$
f_3	Frequent words (fw)	$\frac{\sum_{fw \in s_m} Count(gram_N)}{\lvert fw \rvert . \lvert s_m \rvert}$
f_4	Sentence position	$\frac{\lvert n/2 - m \rvert}{n/2}$
f_5	Sentence length	$1 - \frac{\lvert AL(S) - \lvert s_m \rvert \rvert}{\max(\lvert s_m \rvert)}$
f_6	Sentence similarity	$\frac{\sum_{m'=1, m \neq m'}^{n} (Sim(s_m, s_{m'}))}{\lvert s_m \rvert}$
f_7	Numerical data (nd)	$\frac{\sum_{nd \in s_m} Count(gram_N)}{\lvert nd \rvert . \lvert s_m \rvert}$

Table 1: Description of Text features

As a result, every sentence $s_m = [f_1, f_2, \ldots, f_7]$ is converted into a vector of seven elements where each element denotes the score of j^{th} feature of m^{th} sentence in numerical form. Next, we calculate the score of each sentence s_m using fuzzy inference system. This system requires a set of if-then rules to process the data which has been generated through training corpus. As a result, every sentence has been assigned a score and accordingly extracted the sentence for summary generation.

3 Training data for summarization

Data-driven fuzzy rules generation model requires a large corpus of documents with the labels indicating linguistic informations for every text feature and their respective reference summaries. Therefore, we have created an annotated data for training. In this regard, we have extracted 90% of documents from DUC2002 dataset. The 10% of the dataset has been preserved for testing of summarization model. At first, we pre-process the data

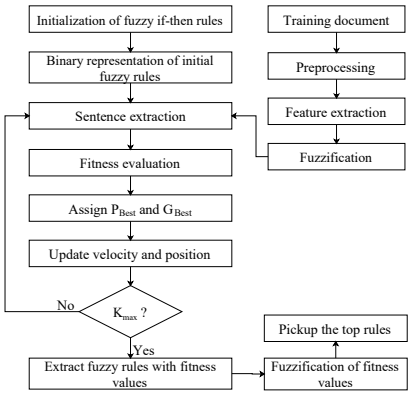


Figure 1: Data-driven fuzzy rules generation

and then compute the scores of text features as given in Table 1. These scores of text features are then used as input for fuzzification process. We use trapezoidal membership function for each input which has been described in Section 5. As a result, every score is represented in the form of linguistic information and every sentence is represented as a set of linguistic informations.

4 Fuzzy evolutionary learning approach

In this work, fuzzy evolutionary learning has been used to find a set of significant fuzzy rules. The fuzzy if-then rules are extracted as linguistic information with the association of fuzzy to discrete PSO algorithm. The flow chart of fuzzy rules generation is illustrated in Figure1.

4.1 Rule encoding

At first, we encode the fuzzy rules. Three linguistic informations (low, medium, and high) have been used to represent each text feature in the rule. These linguistic informations are encoded as 1: Low, 2: $Medium$, and 3: $High$. Therefore, if a rule is, for example, 'If f_1 is Low, f_2 is $Medium$, f_3 is low, f_4 is $High$, f_5 is $High$, f_6 is $High$, f_7 is Low then S_m is $important$ '. This rule can be encoded as 1213331.

4.2 Initialization

The initial encoded form of if-then rules are generated randomly in the range of $[0, 3]$ in the form of population $X = \{x_1, x_2, \ldots, x_p\}$, where $x_q = \{x_{q,1}, x_{q,2}, \ldots, x_{q,dim}\}$ denotes q^{th} rule in the swarm, each element of x_q is denoted by $x_{q,j}$, $q = \{1, 2, \ldots, p\}$ and $j = \{1, 2, \ldots, dim\}$, and population size p. In particular, we generate the

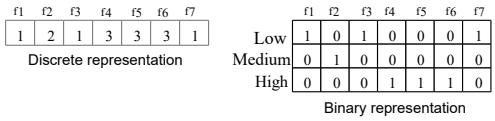


Figure 2: Fuzzy rules representation

rules in the context of text features and seven text features are considered here. Therefore, the dim of every particle is seven. Next, since the elements of every particle is in discrete form as shown in Figure 2, the notion of discrete particle swarm optimization (DPSO) proposed by Izakian et al. (2010) is applied here. In particular, each particle is converted into binary matrix form, as position update with discrete values is an issue in PSO. In the proposed method, every particle is converted into 3×7 matrix as shown in Figure 2. Along with this, the parameters used in DPSO algorithm are also set.

4.3 Sentence extraction

After the initialization of rules, the sentences in the training corpus that follows the rule described in a particle are extracted. There can be any number of sentences that follows a rule. These sentences are then used for evaluation of the rule.

4.4 Fitness function

In this step, we use ROUGE-1 recall function as the fitness function. It calculates the fitness value by matching one to one gram between the system summary and reference summary. It is defined as follows.

$$Fitness(x_q) = \frac{\sum_{s \in Sum_{ref}} \sum_{gram_1 \in s} Count_{match}(gram_1)}{\sum_{s \in Sum_{ref}} \sum_{gram_1 \in s} Count(gram_1)} \quad (1)$$

where Sum_{ref} represents the reference summary. $Count_{match}(gram_1)$ represents the matching grams between the selected set of sentences and the reference summary. The solution having the highest fitness value is marked as the global best solution.

4.5 Particle update

Here, we explain the updation of the particles in the matrix form. Similar to PSO, in DPSO, each element in the matrix of a particle is calculated by the Eq.2 and position of each matrix is updated on the basis of velocity matrix as given in Eq.3.

$$V_q^{k+1}(t,j) = V_q^k(t,j) + c_1 r_1 (pbest_q^k(t,j) - X_q^k(t,j)) + c_2 r_2 (gbest^k(t,j) - X_q^k(t,j)) \quad (2)$$

$$x_q^{k+1}(t,j) = \begin{cases} 1, & \text{if } (V_q^{k+1}(t,j) = \max V_q^{k+1}(t,j)) \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

where $V_q^{k+1}(t,j)$ is the element of the t^{th} row and the j^{th} column of the q^{th} velocity matrix at the $(k+1)^{th}$ iteration. $x_q^{k+1}(t,j)$ is the element of the t^{th} row and the j^{th} column of the q^{th} position matrix at the $(k+1)^{th}$ iteration. c_1 and c_2 are positive acceleration constants, and r_1 and r_2 are random numbers belonging to [0,1]. Here, the position updating equation represents that the value 1 is assigned to those element in a column whose corresponding velocity element has maximum value in that column. If two velocity elements belongs to maximum value in a column then any one has been randomly chosen for assigning the value 1.

4.6 Termination of algorithm

If the number of user-defined iterations is over, all the generated rules in the swarm which are evaluated in the optimization process are extracted with their fitness values. These values are again fuzzified in another class of linguistic information (important, average, and unimportant) in the same manner as given in Section 5. Here, we again use trapezoidal membership function for fuzzification. Finally, all the rules which are identified as important rules are extracted for summary generation. If user-defined iterations is not over then new velocity vector is calculated again to update the velocity using Eqs. 2 and so position matrix.

5 Summary generation

Once the fuzzy rules are generated, the summary of test documents has been generated using fuzzy inference system. Similar to training documents, we pre-process the test documents and extract the features of the sentences. Next, the following process has been done to accomplish the summarization task.

5.1 Fuzzification

Three fuzzy set are considered for fuzzification: low, medium, and high. For every input, say

f_j, we use trapezoidal membership function for fuzzifing values of text features. Each membership function consists of four parameters $(\alpha, \beta, \gamma, \delta)$. With these definitions, a trapezoidal membership function $\mu_{ij}(f_j) \rightarrow [0, 1]$ for i^{th} fuzzy set on the j^{th} input variable can be defined with the condition $\alpha_{ij} \leq \beta_{ij} \leq \gamma_{ij} \leq \delta_{ij}$ as follows.

$$\mu_{ij}(f_j) = \begin{cases} \frac{f_j - \alpha_{ij}}{\beta_{ij} - \alpha_{ij}}, & \text{if } \alpha_{ij} < f_j < \beta_{ij} \\ 1, & \text{if } \beta_{ij} < f_j < \gamma_{ij} \\ \frac{\delta_{ij} - f_j}{\delta_{ij} - \gamma_{ij}}, & \text{if } \gamma_{ij} < f_j < \delta_{ij} \\ 0, & Otherwise \end{cases} \quad (4)$$

5.2 Inference

In this step, the facts resulted in fuzzification process are exploited with the generated data-driven if-then rules to perform the fuzzy reasoning process. We again use trapezoidal membership function for every output which is defined as $\mu'_i(\alpha_i, \beta_i, \gamma_i, \delta_i)$.

5.3 Defuzzification

It is the process of transforming the fuzzy results obtained from inference process into crisp values which has been accomplished using centroid method $Z = \frac{\sum_{j=1}^{q} Z_j u_c(z_j)}{\sum_{j=1}^{q} u_c(z_j)}$ (Sivanandam et al., 2007), where $u_c(Z_j)$ represents the membership in class c (output fuzzy set denotes to class of s_m) at value Z_j (value obtained from fuzzy rules). Hence, the final score of the sentence Z is obtained by their all fuzzy scores and membership degrees. According to these scores, top l sentences are extracted and reordered according to the original document for summary generation.

6 Evaluations and Results

The experiment for evaluating the proposed method is conducted on the DUC2006 (Document Understanding Conference) dataset. This is a benchmark datasets in the field of text summarization that contain the document collections along with reference (human generated) summaries.

The extensively used evaluation methods for summarization system, are known as ROUGE (R) (R-N, R-L, and R-SU), proposed by Lin (2004), has been used in this study. We have computed recall matrix for every evaluation method to show the performance of the proposed method.

We have compared the performance of our method with four other methods: MsWord, NN-SE (Cheng and Lapata, 2016), FEOM (Song et al., 2011), and UniRank (Wan, 2010). MsWord is a benchmark summarizer which is extensively used for summarization. NN-SE is a deep learning based method. FEOM is a fuzzy evolutionary based method in which sentences of document is clustered and elite sentences from each cluster are extracted for summary generation. UniRank is a graph based method which verify the mutual influence between two tasks for text summarization.

Methods	R-1	R-2	R-L	R-SU
Proposed	0.482	0.239	0.401	0.312
MsWord	0.439	0.201	0.382	0.267
NN-SE	0.473	0.233	0.329	0.291
FEOM	0.477	0.224	0.417	0.303
UniRank	0.463	0.231	0.397	0.307

Table 2: Evaluation results on DUC2006

Table 2 shows the comparison between proposed method and other methods using R-1, R-2, R-L, and R-SU on DUC2006 dataset. We observed that the proposed method performed better in the case of R-1, R-2, and R-SU. However, in the case of R-L, FEOM got better result and was outperformed by 3.8%. The proposed method obtains maximum improved results by 18.2% and 16.8% in case of R-2 and R-SU with Msword. NN-SE got second position for R-1. It also perform slightly better than UniRank method in the same case. By observing these results we found that the performance of FEOM is very close to proposed method. So, paired t-test experiment is performed for these methods to find statistical differences between their performances at 5% significance level. We found p-value=0.042 which shows the significant performance of the proposed method. Overall, our proposed model achieves better performance in comparison to other methods.

7 Conclusion

We explore fuzzy swarm intelligence based an effective model for fuzzy rules generation in the context of text summarization. The effective optimization power of PSO bring a surge of interest in this task. The proposed approach considerably outperforms other methods on DUC dataset. For future work, we are planning to enhance our rule generation model by creating a large corpus for its better training. We are also planning to apply our model for multi-document summarization.

References

Razieh Abbasi-ghalehtaki, Hassan Khotanlou, and Mansour Esmaeilpour. 2016. Fuzzy evolutionary cellular learning automata model for text summarization. *Swarm and Evolutionary Computation*, 30:11–26.

Mohammed Salem Binwahlan, Naomie Salim, and Ladda Suanmali. 2010. Fuzzy swarm diversity hybrid model for text summarization. *Information processing & management*, 46(5):571–588.

MS Binwahlan, N Salim, and L Suanmali. 2009. Intelligent model for automatic text summarization. *Information Technology Journal*, 8(8):1249–1255.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*.

Harold P Edmundson. 1969. New methods in automatic extracting. *Journal of the ACM (JACM)*, 16(2):264–285.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479.

Rafael Ferreira, Luciano de Souza Cabral, Rafael Dueire Lins, Gabriel Pereira e Silva, Fred Freitas, George DC Cavalcanti, Rinaldo Lima, Steven J Simske, and Luciano Favaro. 2013. Assessing sentence scoring techniques for extractive text summarization. *Expert systems with applications*, 40(14):5755–5764.

Hesam Izakian, Behrouz Tork Ladani, Ajith Abraham, Vaclav Snasel, et al. 2010. A discrete particle swarm optimization approach for grid job scheduling. *International Journal of Innovative Computing, Information and Control*, 6(9):1–15.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 71–78. Association for Computational Linguistics.

Inderjeet Mani. 2001. *Automatic summarization*, volume 3. John Benjamins Publishing.

Masrah Azrifah Azmi Murad and Trevor Martin. 2007. Similarity-based estimation for document summarization using fuzzy sets. *International Journal of Computer Science and Security*, 1(4):1–12.

Ani Nenkova, Lucy Vanderwende, and Kathleen McKeown. 2006. A compositional context sensitive multi-document summarizer: exploring the factors that influence summarization. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 573–580. ACM.

SN Sivanandam, Sai Sumathi, SN Deepa, et al. 2007. *Introduction to fuzzy logic using MATLAB*, volume 1. Springer.

Wei Song, Lim Cheon Choi, Soon Cheol Park, and Xiao Feng Ding. 2011. Fuzzy evolutionary optimization modeling and its applications to unsupervised categorization and extractive summarization. *Expert Systems with Applications*, 38(8):9112–9121.

Xiaojun Wan. 2010. Towards a unified approach to simultaneous single-document and multi-document summarizations. In *Proceedings of the 23rd international conference on computational linguistics*, pages 1137–1145. Association for Computational Linguistics.

Hongyuan Zha. 2002. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 113–120. ACM.

A Content-based Recommendation System for Medical Concepts: Disease and Symptom

Anupam Mondal **Dipankar Das** **Sivaji Bandyopadhyay**
Department of Computer Science and Engineering
Jadavpur University, Kolkata, India
link.anupam@gmail.com, dipankar.dipnil2005@gmail.com,
sivaji_cse_ju@yahoo.com

Abstract

Dealing with healthcare data is becoming difficult because decision-making becomes crucial to extract information from a huge volume of medical concepts being evolved on daily basis. Moreover, unstructured and semi-structured medical corpora and lack of domain-experts fueled more challenges in this research arena. In order to face one of such challenges, we have developed a baseline model of Medical Recommendation System (MRS). Primarily, MRS helps the experts (e.g. medical practitioners and doctors) by suggesting relevant diseases and symptoms as well as their in-between similarities. Here, we have used a content-based approach to identify similar types of diseases and symptoms by employing two well-known distance metrics, Manhattan and Euclidean. Evaluation based on perplexity score reveals that the performance of MRS is equally well for identifying relevant diseases and symptoms.

1 Introduction

During last few decades, medical information retrieval and extraction behavior are largely observed in the web. A recent survey says that 59% and 49% of U.S. and Indian internet users [1] are looking for online health information e.g., diseases, diagnosis, and treatments (Fischer et al., 2014). Such information helps the doctors as well as patients in their decision-making process for treatment.

Besides, medical experts face difficulties in identifying relevant information from the web due to information overloading (Sommerhalder et al., 2009). In order to overcome such challenges, various domain-specific information extraction systems are essential to help personalized delivery by identifying relevant information (Roitman et al., 2010).

In the present task, we have developed a Medical Recommendation System (MRS), an information extraction system that assists in recommending similar type of diseases as well as symptoms with respect to a particular symptom and disease, respectively. Therefore, we have employed two similarity matrices, disease and symptom. The disease similarity matrix contains similar diseases which have common symptoms, whereas symptom similarity matrix presents similar symptoms with respect to common diseases.

In order to develop the similarity matrices and prepare a disease-symptom relational matrix, we have employed WordNet of Medical Events (WME 3.0) (Mondal et al., 2016), a domain-specific lexicon. Thereafter, the similarity matrices and two well-known distance measurement techniques namely Manhattan and Euclidean have been used to build the proposed MRS. Additionally, we have observed the following challenges to design this recommendation system.

A. *How to identify the categories of disease and symptom for medical concepts?*

B. *How to detect the relation between diseases and symptoms?*

C. *How to frame matrices of similar diseases and symptoms?*

D. *How to recommend the disease and symptom based on the number of user-provided symptoms and diseases, individually?*

E. *How to evaluate the proposed MRS?*

In order to address these challenges, we have employed WME 3.0 lexicon and two well-known similarity measurement techniques such as Man-

[1] http://www.prmoment.in/category/pr-news/survey-shows-that-49-of-indians-use-the-internet-for-health-information

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 120–126

hattan and Euclidean distance. Additionally, we have prepared a disease-symptom matrix in the presence of WME 3.0 lexicon and a Healthline resource [2]. Finally, we have applied Latent Semantic Indexing (LSI) method on the disease-symptom matrix to identify the hidden relations between them.

2 Background of the Work

2.1 Medical Concepts and their Categories Assignment

The research on biomedical information extraction is demanding to extract medical concepts and their relations from the daily produced large amount of unstructured and semi-structured medical corpora. In order to present a structured corpus and extract subjective information from corpora, we have observed that the domain-specific ontologies and lexicons are essential (Borthwick et al., 1998). To this end, the standard vocabularies and ontologies, namely UMLS (Unified Medical Language System), GATE (General Architecture for Text Engineering), and SNOMED-CT (Systematized Nomenclature of Medicine-Clinical Terms), and lexicons namely MEN (Medical WordNet) and WME (WordNet of Medical Event) were used by the researchers (Smith and Fellbaum, 2004; Kilgarriff and Fellbaum, 2000; Chaturvedi et al., 2017; Mondal et al., 2015, 2016).

These ontologies and lexicons help to extract the relevant information from the corpus such as medical concepts, their categories, and relations between them. Besides, the medical terms or concepts extraction from a clinical corpus is treated as an ambiguous task (Styler IV et al., 2014). A group of researchers introduced a sense selection and pruning strategy to expand the ontology in the medical domain (Widdows et al., 2006).

Eklund (Eklund, 2011) developed an annotation system to extract the relations as *diseases* for *treatments* from the scientific medical corpus. Yao, et al. (Yao et al., 2010) extracted relations such as *cures, prevents, and side effects*, which describe the distinctive nature of the biomedical text (medical papers) (Abacha and Zweigenbaum, 2011; Frunza and Inkpen, 2010). Franzen et al. (Franzén et al., 2002) have annotated Yapex corpus with 200 medical abstracts to extract the category as *proteins*. These ontologies are fundamentally looking for extracting *protein-protein* interaction and *disease-treatment* relations from corpora under a BioText project (Rosario and Hearst, 2005).

[2] http://www.healthline.com/

2.2 Recommendation System

Since last decades, recommendation systems are attracting in healthcare services along with the online shopping systems (Ricci et al., 2015; Adomavicius and Tuzhilin, 2005). The recommendation system provides support to extract the relevant and novel information from the corpus and increases the diversity of recommendation.

Adomavicius and Tuzhilin (Adomavicius and Tuzhilin, 2005) generalized the recommendation problem as a utility function u: C x S $->$ R, where C is the set of users and S is the set of recommendable items. u(C, S) returns a real value $>= 0$, where larger values presume a higher interest of C in S and u(C, S) = 0 presumes no interest of C in S. Initially, u is only a partially defined function, where known item ratings are given via users' profiles.

In order to build a recommendation system and compute u(C, S), primarily three approaches are usually followed such as content-based, collaborating-filtering, and hybrid. Content-based approach presents a sparse matrix according to the liking and disliking of the items of the user (Balabanović and Shoham, 1997). On the other side, the collaborating-filtering works with item-item and user-user similarity matrix based techniques with respect to the rating of the users (Cheung and Tian, 2004). The hybrid approach helps to combine the above-mentioned two approaches in an effective way for improving the accuracy (Zhang et al., 2017).

2.3 Medical Recommendation System

Recommendation is a useful technique that helps to find the relevant item for the users. Primarily, we have observed that the recommendation system is used to overcome the information overloading challenges with various type of items such as books, movies, and medical conditions namely diseases and treatments. In the present work, we have developed a Medical Recommendation System (MRS) to recommend subjective information from the textual content in healthcare services (Paruchuri, 2016).

In connection to MRS, Eysenbach and Jadad (Eysenbach and Jadad, 2001) developed a healthcare recommendation system to link the personal online accessible health records

with general health information from evidence-based resources. On the other hand, Roitman et al. (Roitman et al., 2010) observed the personalized recommendation, a valid approach to increase patient safety by avoiding so-called adverse drug reactions (ADR) (Wiesner and Pfeifer, 2014). We have also noticed that the content recommendation merely supplies medical information such as diseases and symptoms from the web (Agarwal et al., 2013; Wendel et al., 2013).

The earlier mentioned study motivates to build a medical recommendation system for diseases and symptoms using a content-based approach in this research.

3 Dataset Preparation

This sub-section presents, how we have prepared an experimental dataset which helps to build the proposed Medical Recommendation System (MRS). In order to start with, we collected the medical corpus from two different resources namely SemEval-2015 Task-6 [3] and MedicineNet [4]. Initially, we have converted all the acquired texts from the resources into context, which refers each sentence in a corpus. We collected 3647 number of medical contexts from both of the resources and prepared an experimental dataset with 2624 number of unique medical contexts.

Thereafter, we have applied a well-defined medical concept identification system developed by Mondal et. al. (Mondal et al., 2016) to identify medical concepts from contexts. Thereafter, to assign the categories of medical concepts, we have employed an auto-categorization technique developed by Mondal et. al. (Mondal et al., 2017). They have annotated the medical concepts into five different categories (*diseases, symptoms, drugs, human_anatomy*, and *Miscellaneous Medical Terms (MMT)*, an unspecified and undetectable category). Among all these categories, we have selected only two primary frequent categories of medical concepts such as diseases and symptoms for the current research.

On the other hand, we have used healthLine [5] resource to recognize the relationship between the assigned diseases and symptoms in a context for our experimental dataset. The relations help to prepare a disease-symptom matrix, which contains 5069 and 1124 number of unique diseases and symptoms individually. The disease-symptom matrix assists in designing disease-disease and symptom-symptom matrices to recommend similar diseases and symptoms for a particular disease and symptom, respectively. These matrices are processed through a content-based approach for building the proposed MRS system.

[3] http://alt.qcri.org/semeval2015/task6/

[4] http://www.medicinenet.com/script/main/hp.asp

[5] http://www.healthline.com/

4 MRS Implementation

In order to implement the system, the primary required recommended information are similar diseases and symptoms according to the user-provided diseases and symptoms, individually (Mondal et al., 2018). Hence, we have prepared one disease-disease and another symptom-symptom similarity matrix from our experimental dataset. Thereafter, we have employed Manhattan and Euclidean distance techniques as a part of the content-based approach to design the MRS. MRS has been presented by two different types of recommendation systems namely RSDS (recommendation for similar diseases and symptoms) and RDS (Recommendation based on diseases and symptoms). RSDS provides the similar type of diseases and symptoms with respect to a particular disease and symptom consequently. On the other hand, RDS presents common diseases as well as symptoms for a number of symptoms and diseases supplied by users, individually. Both of the recommendation systems under MRS have been illustrated in the following subsections.

4.1 Recommendation for Similar Diseases and Symptoms (RSDS)

In order to recognize similar diseases as well as symptoms for a particular disease and symptom individually, we have developed a content-based approach with the help of Euclidean and Manhattan distance technique. Both of the techniques have been applied to the disease-symptom (Di-Sy) matrix for obtaining disease-disease (Di-Di) and symptom-symptom (Sy-Sy) matrices, respectively. Initially, the Di-Sy matrix presents relevant (score 1) and non-relevant (score 0) between a disease and a symptom. The scores have been assigned through the knowledge-based relation between them as mentioned in our experimental dataset. Unfortunately, we have observed

that the scores are not assigning any partial relations between them due to versatile nature of medical concepts. Hence, we have used two different types of distance measurement techniques namely Euclidean and Manhattan to assign the fractional relations between diseases and symptoms. In the following paragraphs, we have illustrated, how Euclidean and Manhattan distances have been used to calculate the score.

Euclidean Distance: Euclidean distance refers to the straight-line distance between two points in Euclidean space (Greenacre and Primicerio, 2008). In this research, we have represented Di-Sy matrix as a Euclidean space where diseases and symptoms appear as points. Besides, we have identified similar diseases as well as symptoms based on similar symptoms and diseases respectively, which presented as a content-based recommendation system.

We have observed that the Euclidean distance does not provide an adequate accuracy due to the high dimension of disease and symptom vectors (Charulatha et al., 2013). Therefore, we have employed Manhattan distance to overcome the mentioned challenge and improve the accuracy.

Manhattan Distance: Manhattan distance function computes the distance between two items by summing up the differences of their corresponding components (Madhulatha, 2012). The Manhattan distance helps to prepare another set of Di-Di and Sy-Sy matrix to develop the proposed RSDS system.

Thereafter, we have combined Euclidean distance (ED) and Manhattan distance (MD) for both diseases as well as symptoms using equation 1 to identify the similar diseases and symptoms. We have selected a threshold value as $>$ 3.00 to recognize the similar diseases and symptoms for a provided disease and symptom under RSD.

$$Similarity_S = (w_1 * ED) + (w_2 * MD) \quad (1)$$

where w_1 = 0.8 and w_2 = 0.2 present the weight for both of the techniques, individually.

On the other hand, the following subsection describes the development steps of another type of recommendation system namely disease recommendation using various symptoms and symptom recommendation using various diseases.

4.2 Recommendation based on Diseases and Symptoms (RDS)

In case of designing recommendation systems in healthcare, we have observed that the identification of a particular symptom or disease is very difficult with respect to a specific disease or symptom, individually. Hence, we have developed a recommendation system that identifies common symptoms based diseases as well as common diseases based symptoms as suggested by a group of medical practitioners. These assumptions offer an adequate accuracy for the proposed MRS.

Thereafter, the following algorithm assists in recommending the common diseases for a particular set of symptoms and vice-versa.

Step-1: Initially, we have presented symptom vectors as SV_{Di-Sy} respect to all diseases from Di-Sy matrix.
Step-2: Take n number of input symptoms (S_i) and generate their corresponding symptom vectors (SV_i).
Step-2.1: If $SV_i \in SV_{Di-Sy}$:

$$SV_i =< a_1, ..., a_{5069} >,$$

where a refers 0 or 1.
Step-2.2: Else:

$$SV_i =< b_1, ..., b_{5069} >$$

where b presents only 0.
Step-3: Common diseases for all n number of symptoms present by CD vector.

$$CD = \bigcup_{k=1}^{n} SV_k \quad (2)$$

Step-4: CD vector helps to recommend common diseases based on the value 1.

5 Evaluation

In order to validate both the recommendation systems under MRS, we have used perplexity distribution approach on Di-Sy matrix, which has been treated as a baseline. Perplexity presents a measurement of how well a probability distribution or probability model predicts a sample in information theory. Equation 3 defines the perplexity of a discrete probability distribution (PD).

$$PD = 2^{\tilde{H}_r} \quad \text{where} \quad \tilde{H}_r = -\frac{1}{T}\log_2 p(w_1, \ldots, w_T) \quad (3)$$

where $\{w_1, \ldots, w_T\}$ is held out test data that provides the empirical distribution $q(\cdot)$ in the cross-entropy using equation 4.

$$\tilde{H} = -\sum_{x} q(x) \log p(x) \tag{4}$$

and $p(\cdot)$ is the recommended system estimated on a training set.

$$H(X) = E[-\log(p(x))] \tag{5}$$

$$\tilde{H} = -\sum_{x} q(x) \log p(x) \tag{6}$$

Perplexity provides a score of difficulty label of the prediction problem, where information entropy [6] measures the unpredictability. Equation 5 and equation 6 refer the entropy (H(X)) of random variable X for linear and discrete domain individually. These equations help to calculate the perplexity score for the different set of diseases as well as symptoms of Di-Sy matrix. Table 1 shows the distribution of perplexity scores for all sets of diseases over symptoms that are initially indicated as Di-Sy matrix and vice-versa.

# Diseases	**# Symptoms**	**Perplexity Score**
5069 (Overall)	1124	**283.50**
1-1267 (First Quarter)	1124	*107.69*
1268-2535 (Second Quarter)	1124	109.74
2536-3802 (Third Quarter)	1124	110.94
3803-5069 (Fourth Quarter)	1124	110.00
# Symptoms	**# Diseases**	**Perplexity Score**
1124 (Overall)	5069	**86.00**
1-281 (First Quarter)	5069	36.21
282-562 (Second Quarter)	5069	34.61
563-843 (Third Quarter)	5069	*33.86*
844-1124 (Fourth Quarter)	5069	36.89

Table 1: A detailed statistics of perplexity scores for various combination of diseases over symptoms and vice-versa.

		Symptoms		
		Ear-Discharge	Breast-Pain	Clubfoot
Diseases	asthma	0	0	0
	HIV	0	0	0
	Lung Cancer	1	1	1
	Pneumonia	1	1	1
	Narcolepsy	0	0	0
	SVD for LSI Score	3.039	2.183	1.414

Table 2: A sample LSI output of the disease-symptom matrix under MRS.

In addition to validate the output of the proposed MRS, we have applied Latent Semantic Indexing (LSI) method. It helps to discover the hidden relation between diseases and symptoms from the baseline Di-Sy matrix. Each disease and symptom is presented as a vector with elements corresponding to these symptoms and diseases, individually. Each element in a vector refers to the weighted association between the concepts as the category of diseases and symptoms. This method assists in describing the efficiency of the prepared Di-Sy matrix in the process of developing MRS along with Singular Value Decomposition (SVD) technique [7]. Table 2 shows a sample output of the disease - symptom matrix of MRS using the BlueBit calculator [8].

The result indicates the developed RDS provides a better prediction over RSDS due to the structure of our experimental dataset.

6 Conclusion and Future Work

In this article, we have attempted to build a medical recommendation system (MRS) using a content-based approach to better services in healthcare. Our primary motivation behind this research is to help the medical experts and non-experts to understand the domain-specific knowledge and their in-between relations. So, we have distributed the overall task into four sub-tasks such as 1) experimental dataset preparation, 2) a relational matrix building namely disease-symptom (Di-Sy), 3) development of similar diseases and symptoms recommendation system (RSDS), and 4) symptoms based diseases and diseases based symptoms recommendation system (RDS).

In order to prepare the experimental dataset and baseline matrix, we have employed WME 3.0, a domain-specific lexicon, Healthline [9] resource. Thereafter, Euclidean and Manhattan distance techniques have been applied to various disease and symptom vectors as content-based recommendation system to build both RSDS and RDS systems.

In future, we will try to improve the accuracy of the proposed MRS by enriching the experimental dataset. We will also focus on design a ranking based technique viz. collaborative filtering instated of the applied content-based to recommend the adequate output for the MRS.

[6] http://pespmc1.vub.ac.be/ENTRINFO.html

[7] http://webhome.cs.uvic.ca/ thomo/svd.pdf

[8] http://www.bluebit.gr/matrix-calculator

[9] http://www.healthline.com/

References

Asma Ben Abacha and Pierre Zweigenbaum. 2011. A hybrid approach for the extraction of semantic relations from medline abstracts. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 139–150. Springer.

Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749.

Deepak Agarwal, Bee-Chung Chen, Pradheep Elango, and Raghu Ramakrishnan. 2013. Content recommendation on web portals. *Communications of the ACM*, 56(6):92–101.

Marko Balabanović and Yoav Shoham. 1997. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72.

Andrew Borthwick, John Sterling, Eugene Agichtein, and Ralph Grishman. 1998. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In *Proc. of the Sixth Workshop on Very Large Corpora*, volume 182.

BS Charulatha, Paul Rodrigues, T Chitralekha, and Arun Rajaraman. 2013. A comparative study of different distance metrics that can be used in fuzzy clustering algorithms. *International Journal of Emerging Trends and Technology in Computer Science (IJETTICS)*.

Iti Chaturvedi, Edoardo Ragusa, Paolo Gastaldo, Rodolfo Zunino, and Erik Cambria. 2017. Bayesian network based extreme learning machine for subjectivity detection. *Journal of The Franklin Institute*.

Kwok-Wai Cheung and Lily F Tian. 2004. Learning user similarity and rating style for collaborative recommendation. *Information Retrieval*, 7(3-4):395–410.

Ann-Marie Eklund. 2011. Relational annotation of scientific medical corpora. In *LOUHI 2011 Third International Workshop on Health Document Text Mining and Information Analysis*, page 27.

Gunther Eysenbach and Alejandro R Jadad. 2001. Evidence-based patient choice and consumer health informatics in the internet age. *Journal of medical Internet research*, 3(2).

Shira H Fischer, Daniel David, Bradley H Crotty, Meghan Dierks, and Charles Safran. 2014. Acceptance and use of health information technology by community-dwelling elders. *International journal of medical informatics*, 83(9):624–635.

Kristofer Franzén, Gunnar Eriksson, Fredrik Olsson, Lars Asker, Per Lidén, and Joakim Cöster. 2002. Protein names and how to find them. *International journal of medical informatics*, 67(1):49–61.

Oana Frunza and Diana Inkpen. 2010. Extraction of disease-treatment semantic relations from biomedical sentences. In *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing*, pages 91–98. Association for Computational Linguistics.

Michael Greenacre and R Primicerio. 2008. Measures of distance between samples: Euclidean. *Fundacion BBVA Publication (December 2013). ISBN*, pages 978–84.

Adam Kilgarriff and Christiane Fellbaum. 2000. Wordnet: An electronic lexical database.

T Soni Madhulatha. 2012. An overview on clustering methods. *arXiv preprint arXiv:1205.1117*.

Anupam Mondal, Erik Cambria, Dipankar Das, Amir Hussain, and Sivaji Bandyopadhyay. 2018. Relation extraction of medical concepts using categorization and sentiment analysis. *Cognitive Computation*, pages 1–16.

Anupam Mondal, Erik Cambria, Antonio Feraco, Dipankar Das, and Sivaji Bandyopadhyay. 2017. Auto-categorization of medical concepts and contexts. In *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on*, pages 1–7. IEEE.

Anupam Mondal, Iti Chaturvedi, Dipankar Das, Rajiv Bajpai, and Sivaji Bandyopadhyay. 2015. Lexical resource for medical events: A polarity based approach. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 1302–1309. IEEE.

Anupam Mondal, Dipankar Das, Erik Cambria, and Sivaji Bandyopadhyay. 2016. Wme: Sense, polarity and affinity based concept resource for medical events. *Proceedings of the Eighth Global WordNet Conference*, pages 242–246.

Venkata A Paruchuri. 2016. Med-hyrec: A recommendation system for medical domain. In *Information Systems Design and Intelligent Applications*, pages 605–611. Springer.

Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B Kantor. 2015. *Recommender systems handbook*. Springer.

Haggai Roitman, Yossi Messika, Yevgenia Tsimerman, and Yonatan Maman. 2010. Increasing patient safety using explanation-driven personalized content recommendation. In *Proceedings of the 1st ACM International Health Informatics Symposium*, pages 430–434. ACM.

Barbara Rosario and Marti A Hearst. 2005. Multi-way relation classification: application to protein-protein interactions. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 732–739. Association for Computational Linguistics.

Barry Smith and Christiane Fellbaum. 2004. Medical wordnet: a new methodology for the construction and validation of information resources for consumer health. In *Proceedings of the 20th international conference on Computational Linguistics*, page 371. Association for Computational Linguistics.

Kathrin Sommerhalder, Andrea Abraham, Maria Caiata Zufferey, Jürgen Barth, and Thomas Abel. 2009. Internet information and medical consultations: experiences from patients and physicians perspectives. *Patient education and counseling*, 77(2):266–271.

William F Styler IV, Steven Bethard, Sean Finan, Martha Palmer, Sameer Pradhan, Piet C de Groen, Brad Erickson, Timothy Miller, Chen Lin, Guergana Savova, et al. 2014. Temporal annotation in the clinical domain. *Transactions of the Association for Computational Linguistics*, 2:143–154.

Sonja Wendel, Benedict GC Dellaert, Amber Ronteltap, and Hans CM van Trijp. 2013. Consumers intention to use health recommendation systems to receive personalized nutrition advice. *BMC health services research*, 13(1):126.

Dominic Widdows, Adil Toumouh, Beate Dorow, Ahmed Lehireche, et al. 2006. Ongoing developments in automatically adapting lexical resources to the biomedical domain. In *Fifth International Conference on Language Resources and Evaluation, LREC, Genoa, Italy*.

Martin Wiesner and Daniel Pfeifer. 2014. Health recommender systems: concepts, requirements, technical basics and challenges. *International journal of environmental research and public health*, 11(3):2580–2607.

Lin Yao, Cheng-Jie Sun, Xiao-Long Wang, and Xuan Wang. 2010. Relationship extraction from biomedical literature using maximum entropy based on rich features. In *2010 International Conference on Machine Learning and Cybernetics*, volume 6, pages 3358–3361. IEEE.

Yin Zhang, Min Chen, Dijiang Huang, Di Wu, and Yong Li. 2017. idoctor: Personalized and professionalized medical recommendations based on hybrid matrix factorization. *Future Generation Computer Systems*, 66:30–35.

A Deep Learning Model for Event Extraction and Classification in Hindi for Disaster Domain

Zishan Ahmad
Dept. of Computer Sc.and Engg.
Indian Institue of Technology Patna
Patna, Bihar-801106, India
1821cs18@iitp.ac.in

Sovan Kumar Sahoo
Dept. of Computer Sc.and Engg.
Indian Institue of Technology Patna
Patna, Bihar-801106, India
sovan.pcs17@iitp.ac.in

Asif Ekbal
Dept. of Computer Sc.and Engg.
Indian Institue of Technology Patna
Patna, Bihar-801106, India
asif@iitp.ac.in

Pushpak Bhattacharyya
Dept. of Computer Sc.and Engg.
Indian Institue of Technology Patna
Patna, Bihar-801106, India
pb@iitp.ac.in

Abstract

Relevant information extraction in disaster situation plays an important role for crisis management. In this paper we propose a deep learning based method for event extraction in Hindi from the man-made and natural disaster related texts. The overall task is to identify the event triggers from text and then classify them into predefined categories of interest. Our proposed model follows an ensemble architecture where we use Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (Bi-LSTM) network as the base learning models. We crawl the data from various newswire sources, define the event types and its annotation guidelines, annotate the datasets and then create a benchmark setup for event extraction. Experiments on 5-fold cross-validation with approximately 80K token datasets show the macro average and micro average F1-scores of 0.40 and 0.59, respectively for event trigger detection and classification.

1 Introduction

Event extraction is a very important task in Natural Language Processing. Event is a basic unit of knowledge representation, and has been drawing growing attention of the researchers and practitioners as an effective mean for information organization. Event is an occurrence that happens at a particular place and at a particular time or time interval. Different agencies and individuals introduce a large amount of data in the web related to any particular event, by publishing news reports in platforms like personal blogs, web sites and news portals. The amount of data is tremendous and hence retrieving relevant information through manual process is infeasible. Hence, there is a necessity to build robust systems that would be able to mine the desired information in an automated way. Extracting event triggers, classifying them into a predefined set of categories and then associating arguments to these events plays an important role in building an information extraction system. In some of the domains such as disaster, extracting relevant information in appropriate time is very important, in order to alert both, the public and the government. Armed with this effective information, post disaster management activities can be carried out because it not only seeks public attention but also the attention of government as well as non-government agencies which are the key players in post disaster management. Some of the existing works focusing on English language are reported in (Nugent et al., 2017; Dittrich and Lucas, 2014; Yun, 2011; Klein et al., 2013; Burel et al., 2017; Tanev et al., 2008). In contrast, there has not been any significant attempt to build event extraction system in Indian languages. In recent times, some of the works as reported in (Kuila and Sarkar, 2017; Singh et al., 2017) focus event extraction in Indian languages such as Hindi, Tamil and Malayalam. These were mostly focused on extracting events from disaster re-

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 127–136

lated tweets. Where as the first paper used deep learning techniques to solve the problem, later one used classical machine learning techniques such as Support Vector Machine (SVM), Gradient Boosting and Random Forest.

In our current work we propose a deep neural network based model for event extraction that operates in two steps, *viz.* identification of event triggers from text and then classifying them into a set of predefined categories of interest. Our focus is on Hindi language.

1.1 Problem Definition and Contributions

Given a Hindi sentence of the form w_1,w_2,w_3,....w_n, the task is to (i). identifying event triggers from text and (ii). classifying event triggers into a set of predefined categories.

(i) Predict the sequence of labels of the form l_1,l_2,l_3,....l_n, where each label l_i corresponds to w_i and $l_i \in$ I,O,B. The tags I,O and B indicate the inside, outside and beginning of entity of an event [1]. The example mentioned below depicts the input sentence and output label sequence. Event triggers are boldfaced in input sentence example.

- **Input Hindi Sentence:** गृह मंत्रालय मुंबई के **बम विस्फोटों** के मद्देनजर इस बात की विशेष तौर पर जांच कर रहा है कि अक्षरधाम मंदिर और १९९३ के मुंबई **बम विस्फोटों** के फैसलों की प्रतिक्रिया के रूप में तो यह **हमले** नहीं हुए

- **Transliteration:** grih mantraalay mumbai ke **bam visphoton** ke maddenajar is baat kee vishesh taur par jaanch kar raha hai ki aksharadhaam mandir aur 1993 ke mumbai **bam visphoton** ke phaisalon kee pratikriya ke roop mein to yah **hamale** nahin hue

- **Gloss:** home/ ministry/ mumbai/ of/ **bomb**/ **blasts**/ of/ `in_wake_of`/ this/ talk/ of/ special/ modus/ on /investigation /do /stay /is / that /akshardhaam /temple /and /1993 /of /mumbai /**bomb** /**blasts** /of /decisions /of /reaction/ of /form / in/ so/ this/ **attack**/ not/ happened

- **Translation:** In view of the Mumbai **bomb blasts**, the Home Ministry is specially investigating the fact that these **attacks** did not take place as response to the Akshardham Temple and the 1993 Bombay **bomb blasts**.

- **Output:** O O O O I I O O O O O O O O O O O O O O O O O O O O O O O I I O O O O O O O O O I O O

(ii) Classify the detected event triggers into predefined event types. For example, in the above Hindi input sentence the boldfaced event triggers belong to `Terrorist_Attack` type.

The key contributions of our proposed work lies in the following:

- Building a deep learning based event extraction system in Hindi for disaster domain. Our proposed model is an ensemble of both Convolution Neural Network (CNN) (Kim, 2014) and Bidirectional Long Short-Term Memory (Bi-LSTM) (Schuster and Paliwal, 1997).

- Creating a benchmark setup for event extraction in Hindi. This may be used as a baseline model for further research towards this direction.

2 Related Works

Event extraction is a well-known problem in Natural Language Processing (NLP). Both the feature based as well as neural network based approaches have been used to solve this problem. Some of the feature based approaches framed the entire event extraction task into two subtasks, and solve each subtask separately (Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011). The main disadvantage of this approach is the error propagation. To overcome this problem (Li et al., 2013) proposed a joint event extraction algorithm which predicts both event triggers and its arguments simultaneously. In (Yang and Mitchell, 2016) both the approach are used i.e

[1] The encoding scheme is according to IOB2, where I indicates the tokens that appear within trigger, B denotes the beginning of a trigger and O denotes the outside of an event trigger. The B is used only when two events of the same type appear in consecutive sequence

extracting event and entities jointly and that also in document level. Chen et al. (Chen et al., 2015) introduced a convolutional neural network (CNN) based word representation model to capture meaningful semantic regularities for words. CNN captures only the most important information in a sentence but may miss other valuable information. So for multiple-event sentences, they proposed a dynamic multi-pooling CNN (DMCNN). DMCNN uses a dynamic multi-pooling layer according to event triggers and arguments, to reserve more crucial information. They reported that their system significantly outperforms other state-of-the-art systems.

Event extraction task has also been addressed in specialized tracks dedicated in the Text Analysis Conference (TAC). Many research groups participated in the workshop and submitted their works covering various methods (classical supervised and deep learning models). We present here a brief survey of the works that focused on deep learning based methods. Frank et. al. submitted a system at TAC KBP 2016 (Mihaylov and Frank, 2016). They built a neural architecture for event trigger detection, event type classification and event realis classification. They used a Bidirectional Long Short-Term Memory (Bi-LSTM) based system for detecting event triggers from text. They carried out experiments with various configurations including using Part-of-Speech (PoS) and dependency label embeddings as additional information into deep neural network. They also performed experiments with short-cuts to the output layer and reported improvement in the performance. The work reported in (Dubbin et al., 2016) implemented two variants of event detection systems. Their first system used manually created rules and a set of a very rich linguistic resources whereas their second system used deep neural networks.

Event extraction in disaster situation is very crucial as it assists in supplying relevant information to the affected people and the various other stakeholders including the government agencies. A real-time news event extraction system was developed by Joint Research Center of the European Commission (Tanev et al., 2008). The developed system could extract violent and disaster events accurately from on-line news though their system was linguistically lightweight. Another study (Yun, 2011) described a rapid event detection system of disaster events and showed how to detect a target event from tweets. They used features like location, time, keywords and frequency of keywords in tweets. A prototype of real-time multilingual natural disaster identification and monitoring system based on Twitter was introduced by (Dittrich and Lucas, 2014). In (Nugent et al., 2017) the authors compared a set of supervised learning methods to event type classification on English news data. Majority of the works have been carried out mostly in English and some other resource-rich languages. In contrast there has been a very little works on event extraction on resource-poor languages such as the Indian ones. A deep learning based system was built by (Kuila and Sarkar, 2017) for event extraction in Indian languages like Hindi, Tamil and Malayalam. The system was built for handling tweets. Event classification and location prediction was done in (Singh et al., 2017) using traditional machine learning techniques. Event extraction in Tamil language was reported in (SharmilaDevi et al., 2017). They extracted features and classified each word or chunk into event and non-event class using SVM. In another work (Sristy et al., 2017) formulated the event extraction task as sequence labeling problem and used Conditional Random Fields(CRF) to extract events.

It is seen that most of the previous works used tweets for event extraction and classification but our focus is on more structured newswire data, which are collected from different on-line news portals in Hindi language. We follow a pipelined approach where event detection and event classification are tackled in two consecutive steps. Each of the modules is based on an ensemble of Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (Bi-LSTM).

3 Methodology

Our overall task consists of detecting the trigger words and phrases from a given sentence, and classifying these into different types. The schematic diagram of the complete system is

shown in the Figure 1.

We use these models in a pipeline to get the final output. Each sentence is passed through the 'Trigger Detection Model', the task of which is to detect trigger expressions denoting the events. Trigger refers to the textual content that corresponds to the tokens denoting the events related information. The detected triggers are then passed through 'Trigger Classification Model' which classifies the triggers into different event types.

3.1 Word Embedding Representation

The proposed system uses word-embedding vectors of size 300 to represent the words as input to the models. The word embeddings are trained using word2vec algorithm (Mikolov et al., 2013) on Hindi Wikipedia dump. The size of corpus is 323M and vocabulary size is 30,393 tokens. However the word-embeddings are pre-trained and downloaded from github page[2]. To represent Part-of-Speech (PoS) tags, we use one-hot encoding representation.

3.2 Trigger Detection Model

We formulate the task of event trigger detection as a sequence labeling problem, i.e. for each token in the sentence we need to decide whether it denotes an event expression or not. Our proposed model is ensemble in nature where the base models are Bi-LSTM and CNN (c.f. Figure 2).

The word embedding of each word is passed through a Bi-LSTM model, and we obtain an output representation for each word. The word embedding of each word is passed through CNN and convoluted features are obtained. The one-hot vector representation of PoS tag of each word is also passed through a separate Bi-LSTM. We obtain the PoS information from the Hindi Shallow Parser [3]. We obtain an output representation of each PoS tag. Finally, all these output representations are concatenated. This concatenated feature is then passed through a Multi Layer Perceptron (MLP) followed by a Softmax layer which computes the probability distribution over the possible tags of I_Event_Trigger or O_Event_Trigger [4].

[2]http://github.com/Kyubyong/wordvectors

[3]http://ltrc.iiit.ac.in/analyzer/hindi/

[4]Here I and O denote the intermediate tokens of an event expression and outside token, respectively.

3.3 Trigger Classification Model

The input to this event trigger classification model is the trigger expression or phrase and the output is a possible class to be assigned to that particular trigger. Our current work is a multi-class classification problem that classifies each trigger into seventeen possible classes. Similar to trigger detection, we also use here an ensemble network model consisting of Bi-LSTM and CNN. The overall schematic diagram of the classification model is depicted in Figure 3. We pad each trigger phrase by the placeholder '<pad>', to make each phrase of equal length. Word embedding of each word is passed through a CNN to obtain the convoluted feature representation. These convoluted features are flattened and concatenated with Bi-LSTM representation. This concatenated feature is then passed through a Multi Layer Perceptron model followed by a Softmax layer, which is used to obtain the probability distribution over a set of seventeen classes.

4 Datasets and Experiments

In this section, we provide the description of the datasets, experimental setup, results and provide necessary analysis of the results.

4.1 Datasets

The news data are crawled from several Hindi news portals. In the dataset there are 253 news documents consisting of 4,403 sentences. In total 80,136 words are present among which 1,179 words are trigger words. All the news documents are annotated by three annotators who are from linguistic background and having sufficient knowledge of the related area, particularly the TAC-KBP event and entity annotation guidelines. In order to measure the inter-annotator agreement ratio, we asked three annotators to annotate 5% of total documents. We found the multi-rater Kappa agreement ratio of 0.85.

The news data crawled, belong to disaster domain. Seventeen disaster event types including both man-made and natural disaster. The event types are: *Terrorist_Attack, Storm, Cyclone, Normal_Bombing, Earthquake, Transport_Hazard, Floods, Land_Slide,*

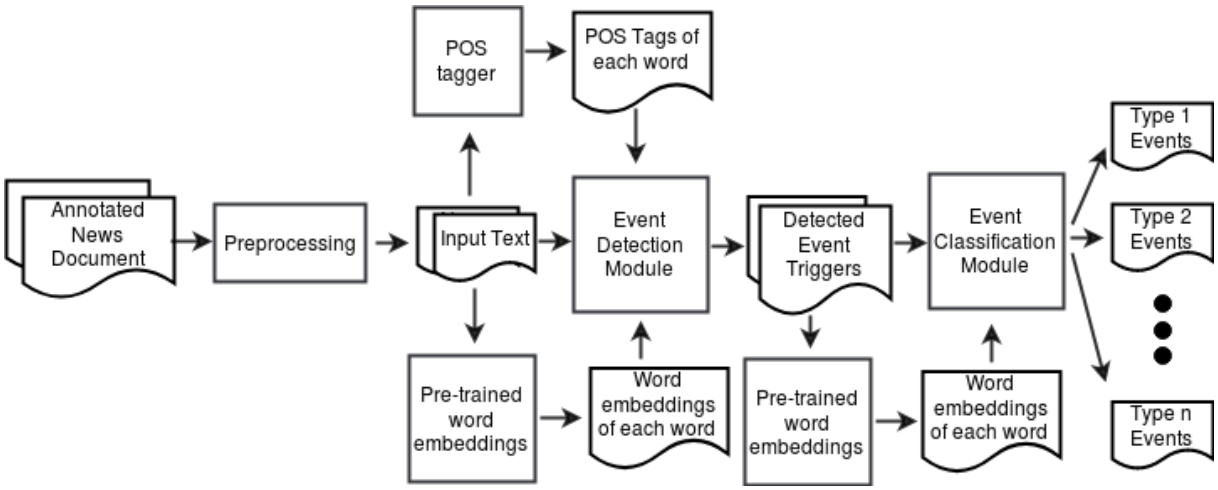


Figure 1: Schematic diagram of the overall system

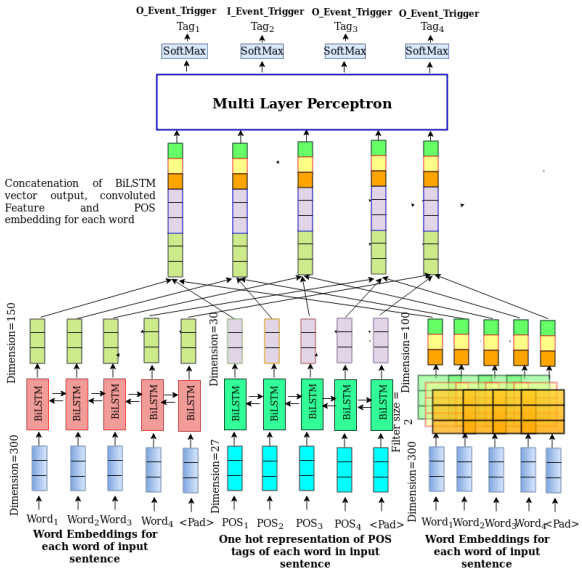


Figure 2: Trigger Detection Model

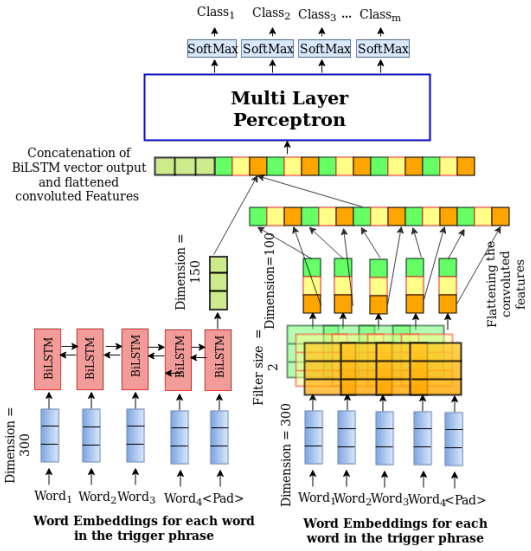


Figure 3: Trigger Classification Model

Train_Collision, Armed_Conflict, Hurricane, Shoot_Out, Riots, Aviation_Hazard, Hail_Storm, Tsunami, Surgical_Strike.

While annotating the event triggers, the above disaster types are used as classes, and each event trigger is associated with one of the seventeen classes. Thus our data contains the event information as well as the fine-grained class information of the event.

4.2 Experimental Setups

For implementing the deep learning models a Python based library Keras [5] is used. We use the IOB format (Ramshaw and Marcus, 1999) [6].

The trigger detection model (c.f. Figure 2) has word-embedding vectors of size 300, and one hot encoding of PoS tags of size 27 as inputs to the Bi-LSTM and CNN layers. The Bi-LSTM layer with word embeddings as input has 150 neurons and the Bi-LSTM layer with encoded PoS tags as input has 20 neurons. The CNN layer has 100 filters sliding over 2 words at a time. *'Relu'* is used as an activation function for CNN and *'dropout'* of 30% is used between layers for regularization. The multi-layer perceptron comprises of 150 and 75 neurons in the first and second layer respec-

[5] http://keras.io

[6] B, I and O denote the beginning, intermediate and outside of an event trigger

tively. *'Softmax'* is used in the final layer for classification of trigger and non-trigger words.

The input to the trigger classification model (c.f. Figure 3) is also word-embedding vectors of size 300. Similar to trigger detection model, Bi-LSTM layer has 150 neurons and CNN layer has 100 filters sliding over 2 words at a time. The *'dropout'* and activation functions used are the same as in the trigger detection model. The multi-layer perceptron comprises of 600 and 100 neurons in the first and second layer respectively and *'Softmax'* is used in the final layer for classification of 17 event classes.

Training is done using a learning rate of 0.001 and *'Adam'* optimizer is used for fast convergence. The data is fed to the neural network in batches of 32. *'Checkpoints'* are used to save the best weights of the model based on training accuracy.

For evaluation we use *'Precision'*, *'Recall'* and *'F1-Score'* as the metrics for event trigger detection. We further use micro and macro averaging of the *precision*,*recall* and *F1-score* of each class for evaluating event classification model. Macro-average computes the metrics independently for each class and then takes the average (hence treating all the classes equally), whereas micro-average aggregates the contributions of all classes to compute the average metric. Thus in case of class imbalance micro-average is a more preferred criteria.

5 Results and Analysis

In this section we present the details of experiments and the results that we obtain along with necessary analysis.

5.1 Trigger Detection Model

For trigger detection we develop three different variations of event trigger detection. First model is based on Bi-LSTM. The word embedding of each word in the sentence is concatenated with one hot vector of its PoS tag. This is then passed through a Bi-LSTM. The output representation of the Bi-LSTM is classified into event and non-event by Softmax at the output layer.

The second model is based on CNN. The input to the CNN is same as that of Bi-LSTM. 100 filters were used to obtain features with a kernel size of 2. The convoluted features, thus obtained, are classified using Softmax at the output layer.

The third model is described in Section 3.2 (Figure 2). We also re-implement the system of event extraction proposed in (Kuila and Sarkar, 2017) and evaluate it on our dataset. Their system uses two back to back CNNs and then a Bi-LSTM in sequence.We demonstrate the results of 5-Fold cross-validation in Table 1.

	Precision	Recall	F1-Score
Bi-LSTM	0.74	0.71	0.72
CNN	0.70	0.69	0.69
Ensemble Model	0.74	0.75	0.74
Kuila et. al, 2017	0.68	0.75	0.71

Table 1: Results of different event trigger detection models: 5-fold cross-validation

From the results in Table 1 it can clearly be seen that the ensemble model performs better than the individual Bi-LSTM and CNN models. The Bi-LSTM model treats the problem as a sequence to sequence labeling task and it tries to classify the current word by looking at the context from both the directions as well as the current input word. The CNN model has a kernel size of 2, i.e. it tries to extract bi-gram features that are relevant in classifying a word into trigger or non-trigger. These two approaches are combined through an ensemble model that exploits both the contextual information from Bi-LSTM and the bi-gram feature extracted from CNN to make prediction. It is able to make use of the strength of both the models and optimize it to solve the problem.

5.2 Trigger Classification Model

For trigger classification we again build three different models. Similar to the trigger detection model we use Bi-LSTM, CNN and the ensemble model (c.f. Figure 3) for trigger classification. We obtain the results in terms of *macro-averaging* and *micro-averaging* of *precision*, *recall* and *F1-score* of each class.

Evaluation results of 5-Fold cross-validation are shown in Table 2. Results show that the ensemble model performs better than the individual models.

We present the detailed class-wise evaluation results in Table 3 for the ensemble model.

	$Macro_p$	$Macro_r$	$Macro_{f1}$
Bi-LSTM	0.40	0.45	0.42
CNN	0.42	0.44	0.43
Ensemble Model	0.45	0.46	0.45
	$Micro_p$	$Micro_r$	$Micro_{f1}$
Bi-LSTM	0.69	0.69	0.69
CNN	0.70	0.70	0.70
Ensemble Model	0.72	0.72	0.72

Table 2: Evaluation results of 5-fold cross-validation for trigger classification. Here, Macro-averaged Precision: $Macro_p$, Macro-averaged Recall: $Macro_r$, Macro-averaged F1-Score: $Macro_{f1}$, Micro-averaged Precision:$Micro_p$, Micro-averaged Recall: $Micro_r$, Micro-averaged F1-Score: $Micro_{f1}$

For the classes for which the performance are very less are due to the less number of instances for the respective class.

Class	Precision	Recall	F1-Score
Terrorist_Attack	0.82	0.95	0.88
Storm	0.72	0.97	0.80
Cyclone	0.67	0.32	0.43
Normal_Bombing	0.20	0.22	0.21
Earthquake	0.44	1.00	0.62
Transport_Hazard	1.00	0.33	0.50
Floods	0.70	1.00	0.82
Land_Slide	1.00	1.00	1.00
Train_Collision	0.02	0.02	0.02
Armed_Conflict	0.01	0.01	0.01
Hurricane	0.00	0.00	0.00
Shoot_Out	0.70	0.95	0.72
Riots	0.40	0.15	0.21
Aviation_Hazard	0.33	0.18	0.23
Hail_Storm	0.02	0.20	0.04
Tsunami	0.20	0.20	0.20
Surgical_Strike	0.20	0.20	0.20

Table 3: Evaluation results of class-wise trigger classification: 5-fold cross-validation. Here we report macro scores.

5.3 Pipelined Model for Trigger Detection and Classification

The final system (c.f Figure 1) uses both event trigger detection model (c.f. Figure 2) and event classification model (c.f. Figure 3). Each document is divided into sentences, and each sentence is represented by the sequence of word-embeddings of its words. This sequence is passed through the trigger detection model and triggers are obtained. These trigger phrases are again represented as a sequence of word-embeddings of its words. The trigger expressions so detected are passed through the trigger classification model, and the appropriate classes of triggers are obtained.

The macro and micro-averaged results of 5-fold cross-validation are shown in Table 4.

	Precision	Recall	F1-Score
Macro-Averaged	0.35	0.48	0.40
Micro-Averaged	0.59	0.59	0.59

Table 4: Evaluation results of 5-fold cross-validation

From Table 4 it can be clearly seen that the performance in this pipelined model is less compared to what are reported in the earlier section. This is because the errors encountered in the trigger detection model are propagated to the next stage, i.e. event classification model. We also show the class-wise evaluation results obtained through 5-fold cross-validation in Table 5.

Since a lot of triggers are not detected by the trigger detection model (i.e. false negative cases), these triggers are not classified at all. A number of false positives in the trigger detection phase also contributes to the overall classification error. These two factors causes a degradation of the overall performance.

Class	Precision	Recall	F1-Score
Terrorist_Attack	0.82	0.97	0.88
Storm	0.40	0.80	0.53
Cyclone	0.51	0.32	0.39
Normal_Bombing	0.77	0.35	0.49
Earthquake	0.63	0.20	0.4
Transport_Hazard	0.89	0.22	0.35
Floods	0.50	0.20	0.28
Land_Slide	0.67	0.32	0.60
Train_Collision	0.20	0.26	0.22
Armed_Conflict	0.01	0.01	0.01
Hurricane	0.00	0.00	0.00
Shoot_Out	0.20	0.61	0.30
Riots	0.36	0.20	0.25
Aviation_Hazard	0.30	0.18	0.22
Hail_Storm	0.02	0.20	0.04
Tsunami	0.1	0.1	0.1
Surgical_Strike	0.2	0.2	0.2

Table 5: Evaluation results of 5-fold cross-validation for pipelined model. Here we report macro scores.

5.4 Error Analysis

We analyze the errors for both trigger detection and trigger classification models. For error analysis we split the data into training set (80% of data) and testing set (20% of data). The training set contains 3,522 sentences and the testing set contains 881 sentences.

5.4.1 Trigger detection model

Out of total triggers of 323 in the test data, 255 triggers are captured by our trigger detection system. Few examples of the errors caused by our system are as follows:

1. The system sometimes confuses between an actual event and a hypothetical event.

   (a) **Sentence-1**: छत्तीसगढ़ में **नक्सली हमले** में 3 जवान शहीद
   Transliteration: Chhatteesagadh mein **naksalee hamale** mein 3 javaan shaheed
   Translation: 3 soldiers killed in **Naxal attack** in Chhattisgarh

   (b) **Sentence-2**: इन आतंकवादियों की दिल्ली में **हमले** की योजना थी
   Transliteration: In aatankavaadiyon kee dillee mein **hamale** kee yojana thee
   Translation: These terrorists had plans to **attack** Delhi

   In the above example, **Sentence-1** talks about a real event but **Sentence-2** talks about a hypothetical event. However, the system is unable to detect the real event in **Sentence-1** but detects the hypothetical event in **Sentence-2**.

2. Long phrases used to describe an event are being confused by the system.

   (a) **Sentence-3** अलबामा में ट्रांसफॅर्मरों **में बारिश का पानी भर गया**
   Transliteration: Alabaama mein traansaphairmaron **mein baarish ka paanee bhar gaya**
   Translation: **Rain water was flooded** in transformers in Alabama

   (b) **Sentence-4** निचले इलाके में मौजूद एक व्यक्ति क्रिस रॅबिंसन ने फोन से बताया कि **पानी तेजी से भर रहा है**
   Transliteration: nichale ilaake mein maujood ek vyakti kris raibinsan ne phon se bataaya ki **paanee tejee se bhar raha hai**
   Translation: Chris Robinson, a person in the lower area, told on the phone that **water is filling up fast**.

   In **Sentence-3** the phrase में बारिश का पानी भर गया describes 'flooding', but this is not captured by the system. However, in **Sentence-4**, the phrase पानी तेजी से भर रहा है is not used for an event, but the system wrongly captures it as an event. This is because both these phrases are very close in meaning, and the context for proper disambiguation is lacking.

3. Another major cause of error is the class-imbalance problem. Out of a total 1,179 trigger words, 543 are of class *Terrorist_Attack*. On the other hand there are only 5 trigger words of type *Tsunami*, 7 triggers of type *Hail_Storm*, 9 triggers of type *Aviation_Hazard* and *Riots*, 14 triggers of type *Shoot_Out* and 15 triggers of type *Hurricane*. Uneven class distribution influences to the overall error. Balancing these classes may reduce such errors.

5.4.2 Trigger classification model

The results of event classification model are discussed in Section 5.2. The confusion matrix after classification (by ensemble model) on the test data can be seen in Figure 4.

	None	Terrorist_Attack	Storm	Cyclone	Normal_Bombing	Earthquake	Transport_Hazard	Floods	Land_Slide	Train_Collision	Armed_Conflict	Hurricane	Shoot_Out	Riots	Aviation_Hazard	Hail_Storm	Tsunami	Surgical_Strike
None	0	46	29	3	9	5	0	2	0	0	1	0	1	0	0	1	0	0
Terrorist_Attack	0	141	0	1	5	0	0	0	0	0	1	0	0	0	0	0	0	0
Storm	0	0	39	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cyclone	0	0	21	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Normal_Bombing	0	7	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
Earthquake	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0
Transport_Hazard	0	0	0	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0
Floods	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Land_Slide	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0
Train_Collision	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Armed_Conflict	0	3	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
Hurricane	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Shoot_Out	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
Riots	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
Aviation_Hazard	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Hail_Storm	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tsunami	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Surgical_Strike	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4: Confusion matrix of classification by event classification model (ensemble model)

From the confusion matrix it can be seen that the system is confused between the types that are very close to each other. *Terrorist_Attack* is being confused with *Normal_Bombing* and *Normal_Bombing* is being confused with *Terrorist_Attack*. We can also see that *Storm* is being confused with *Cyclone*, *Cyclone* is being confused with *Storm* while *Hurricane* is being classified to either *Storm* and *Cyclone*. This is because the events *Storm*, *Cyclone* and *Hurricane* are very close in nature. Also the Hindi word used for all the three is 'तूफान', which further confuses the system.

6 Conclusion and Future Works

In this paper we have presented a hybrid deep learning approach based on Bi-LSTM and CNN for event trigger detection and classification. As there was no readily available data, we have collected Hindi documents from the various web sources, annotated data with event triggers and a predefined set of categories. Preliminary evaluation shows promising results. Experiments on these datasets show that our proposed ensemble model performs better compared to the individual model. This can be used as a benchmark setup for further research and development in the related areas.pr We have also performed detailed analysis to understand the shortcoming of our proposed system. To fix these errors we would like to explore ways in which context could be added to the input. This will help the system in making better informed decision for trigger detection. To mitigate the class imbalance problem, we would like to crawl enough documents relevant to the specific classes that suffer from the data sparsity problem. Another future direction would be to develop a stacked based classifier with Bi-LSTM as a base followed by Conditional Random Field (CRF).

7 Acknowledgement

The research reported here is an outcome of the project titled "A Platform for Cross-lingual and Multi-lingual Event Monitoring in Indian Languages", supported by IMPRINT-1, MHRD, Govt. of India, and MeiTY, Govt. of India.

References

Grégoire Burel, Hassan Saif, Miriam Fernandez, and Harith Alani. 2017. On semantics and deep learning for event detection in crisis situations. *Workshop on Semantic Deep Learning (SemDeep), at ESWC 2017, 29 May 2017, Portoroz, Slovenia*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 167–176.

André Dittrich and Christian Lucas. 2014. Is this twitter event a disaster? *AGILE Digital Editions*.

Greg Dubbin, Archna Bhatia, Bonnie J Dorr, Adam Dalton, Kristy Hollingshead, Suriya Kandaswamy, Ian Perera, and Jena D Hwang. 2016. Improving discern with deep learning. In *TAC*.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1127–1136. Association for Computational Linguistics.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. *Proceedings of ACL-08: HLT*, pages 254–262.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Bernhard Klein, Federico Castanedo, Inigo Elejalde, Diego López-de Ipina, and Alejandro Prada Nespral. 2013. Emergency event detection in twitter streams based on natural language processing. In *International Conference on Ubiquitous Computing and Ambient Intelligence*, pages 239–246. Springer.

Alapan Kuila and Sudeshna Sarkar. 2017. An event extraction system via neural networks. *FIRE (Working Notes)*, pages 136–139.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 73–82.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797. Association for Computational Linguistics.

Todor Mihaylov and Anette Frank. 2016. Aipheshd system at tac kbp 2016: Neural event trigger detection and event type and realis disambiguation with word embeddings. *Proceedings of the TAC Knowledge Base Population (KBP)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Tim Nugent, Fabio Petroni, Natraj Raman, Lucas Carstens, and Jochen L Leidner. 2017. A comparison of classification models for natural disaster and critical event detection from news. In *Big Data (Big Data), 2017 IEEE International Conference on*, pages 3750–3759. IEEE.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

V SharmilaDevi, S Kannimuthu, and G Safeeq. 2017. Kce_dalab@ eventxtract-il-fire2017: Event extraction using support vector machines. *FIRE (Working Notes)*, pages 144–146.

Jyoti Prakash Singh, Yogesh K Dwivedi, Nripendra P Rana, Abhinav Kumar, and Kawaljeet Kaur Kapoor. 2017. Event classification and location prediction from tweets during disasters. *Annals of Operations Research*, pages 1–21.

Nagesh Bhattu Sristy, N Satya Krishna, and Durvasula VLN Somayajulu. 2017. Event extraction from social media text using conditional random fields. In *FIRE (Working Notes)*, pages 140–143.

Hristo Tanev, Jakub Piskorski, and Martin Atkinson. 2008. Real-time news event extraction for global crisis monitoring. In *International Conference on Application of Natural Language to Information Systems*, pages 207–218. Springer.

Bishan Yang and Tom Mitchell. 2016. Joint extraction of events and entities within a document context. *arXiv preprint arXiv:1609.03632*.

Hong-Won Yun. 2011. Disaster events detection using twitter data. *Journal of information and communication convergence engineering*, 9(1):69–73.

Exploring the importance of context and embeddings in neural NER models for task-oriented dialogue systems

Pratik Jayarao
Haptik Inc, Mumbai
pratik.jayarao@haptik.co

Chirag Jain
Haptik Inc, Mumbai
chirag.jain@haptik.co

Aman Srivastava
Haptik Inc, Mumbai
aman.srivastava@haptik.co

Abstract

Named Entity Recognition (NER), a classic sequence labelling task, is an essential component of natural language understanding (NLU) systems in task-oriented dialog systems for slot filling. For well over a decade, different methods from lookup using gazetteers and domain ontology, classifiers over hand crafted features to end-to-end systems involving neural network architectures have been evaluated mostly in language-independent non-conversational settings. In this paper, we evaluate a modified version of the recent state of the art neural architecture in a conversational setting where messages are often short and noisy. We perform an array of experiments with different combinations of including the previous utterance in the dialogue as a source of additional features and using word and character level embeddings trained on a larger external corpus. All methods are evaluated on a combined dataset formed from two public English task-oriented conversational datasets belonging to travel and restaurant domains respectively. For additional evaluation, we also repeat some of our experiments after adding automatically translated and transliterated (from translated) versions to the English only dataset.

1 Introduction

Named Entity Recognition (NER) is a challenging and vital task for slot-filling in task-oriented dialogue models. We define a task-oriented dialogue system where a user and the system take turns exchanging information till some concluding action is performed related to user's query. Any task oriented bot must figure out the correct intent and fill slots for requested task interactively until all required slots required for the related action are filled. For most domains and languages, a very small amount of supervised data is available. Generalizing from such small amount of data can be challenging as for some entities are open ended in what values they can assume like NAME while for some other entities like CITY or LOCATION it is very likely some values will appear very rarely while training. As a result, a lot of hand-crafted features and domain-specific knowledge resources and gazetteers are used for solving this task.

Unfortunately, collecting such resources is time-consuming for each new domain and language making a cold start even harder. Most work for NER has been benchmarked on the popular CoNLL2003 dataset (Sang and Meulder, 2003), OntoNotes 5.0 and few other datasets. Only very recently high quality medium to large sized task-oriented dialogue English datasets like DSTC2 (Henderson et al., 2014), Frames (Asri et al., 2017), etc. have been made available to focus on the challenge of dialogue state tracking. Some of these datasets also happen to have slots tagged hence making them suitable to benchmark NER systems. We combined two such datasets - DSTC2 (Restaurant table booking system) and Frames (Airline ticket booking system) to evaluate our approaches in multi-domain multi-entity setting. We observed that in conversational datasets, the systems usually yield longer informative messages and users often tend to provide information in multiple short messages. For example,

System: What city are you flying to?
User: Paris

In such cases, the immediately previous system utterance provides important context regarding the domain and slots to predict. Users also tend to make spelling mistakes which can create problem for models using words as unit features.

In the past few years, end-to-end neural architectures (Huang et. al., 2015; Lample et al., 2016;

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 137–142

Ma and Hovy, 2016) with a CRF layer have shown promising results for NER. Our work is inspired from these models where we too use an end-to-end BI-LSTM-CRF based tagger network but also include an additional LSTM based context encoder that encodes the system occurrence immediately before the user's query which we use to initialize the tagger network's initial state. We observe that this additional context improves F1 score for all settings we test. Word embedding features (Mikolov et al., 2013; Pennington et al., 2014) obtained using unsupervised training on a large corpora have also shown to improve results (Huang et. al., 2015; Lample et al., 2016; Ma and Hovy, 2016).

Following that we too explore different initializations of word embeddings in our experiments. We also compute additional word representation from character level representations using Convolutional Neural Networks (CNNs) and combine them with pre-trained word embeddings. To validate that our models can work in language-independent settings, we also present results after adding automatically translated (to Hindi) and transliterated (to English back from Hindi) versions of the datasets and find that results follow the same trends as results on the English only dataset.

2 Related Work

NER is viewed as a sequential prediction problem. Most work related to this task if often benchmarked on CoNLL2003 and OntoNotes datasets. Early work includes typical models like HMM (Rabiner, 1989), CRF (Lafferty et al., 2001), and sequential application of Perceptron or Winnow (Collins, 2002). (Ratinov and Roth, 2009) highlight design challenges involved in NER and achieve an F1 of 90.80 on CoNLL2003 using averaged perceptron model trained on non-local features, gazetteers extracted from Wikipedia and brown clusters extracted from large corpora. (Lin and Wu, 2009) get a better score using linear chain CRF with L2 regularization without any gazetteers but instead by obtaining phrase features from clustering a massive dataset of search query logs. (Passos et al., 2014) match their performance by using a linear chain CRF on hand-crafted features and phrase vectors trained using a modified semi-supervised skip-gram architecture. (Luo et al., 2015) jointly model NER and entity linking tasks. They include hand-engineered features like spelling features, lexical clusters, shallow parsing features as well as stemming and large external knowledge bases.

A shift in trend towards more neural based approaches can be traced back to (Collobert et al., 2011b) proposed an effective deep neural model with a CRF layer on top, that requires almost no feature engineering and learns important features from word embeddings trained on large corpora. The model achieved near state of the art results on several natural language tasks including NER. (Santos and Guimaraes, 2015) modify this architecture to incorporate character level features computed using CNNs and report better F1 scores on both CoNLL2003 and OntoNotes. Coming to architectures that involve recurrent neural networks, (Huang et. al., 2015) test LSTM, BI-LSTM, LSTM-CRF and BI-LSTM-CRF networks with word embeddings and hand-crafted spelling and context features for several sequence tagging tasks. (Lample et al., 2016) also use BI-LSTM-CRF models but don't rely on any hand-crafted features and external resources. They use pre-trained word embeddings and character level embeddings computed from another BI-LSTM network as primary features. (Ma and Hovy, 2016) use similar architecture but compute character level embeddings using CNN layer instead of BI-LSTMs. (Chiu and Nichols, 2015) also work with very similar BI-LSTM-CNN-CRF architecture with some extra character level features like character type, capitalization, and lexicon features. Finally, very recent work from (Peters et. al., 2018) show improvements on NER task by computing better contextualized word level embeddings features.

Our work is closely related to (Ma and Hovy, 2016) in terms of the neural architecture such that we too work in an end-to-end setting and include character level features using a CNN. Although we experimented with different recurrent cell types for our models due to space limitations we present only BI-LSTM models. Our work mostly focuses on using conversational context as a source of additional features, determining optimal word embeddings representations for the task.

3 Dataset

We primarily use data from two publicly available dialogue datasets Maluuba Frames (Asri et al., 2017) and Dialog State Tracking Challenge 2 (DSTC2) (Henderson et al., 2014).

To evaluate the system for English we constructed first dataset DSTC-FRAMES-EN by combining the two datasets to get a total of 13599 user-system utterance pairs and 7 entities ('or_city', 'dst_city', 'budget', 'date', 'area', 'food', 'price_range'). We split this data into 12000 train samples and 1599 test samples with a total of 412 unique entities (with IOB-prefixes).

We also check if our models can work with more than one language simultaneously. For this we translate DSTC-FRAMES-EN to Hindi using Google Translate (Wu et al., 2016) and further transliterate this translated version to Latin script using Polyglot transliteration (Chen and Skiena, 2016)**.** We combine all three to form an extended dataset DSTC-FRAMES-ENHI which contains a total of 37785 samples, 7 entities with 1106 unique entities values (with IOB-prefixes). We split this combined dataset into 34000 training samples and 3785 test samples.

4 Model Architecture

4.1 Input Layer

We use two kinds of input representations - word embeddings and character embeddings concatenated with word representations.

Word Embeddings: Word Embeddings have a significant role in the increasing the performance of various neural inspired models as they exploit the syntactic and semantic understanding of a word. We conducted experiments with four different (frozen) embeddings w.r.t. dimension size, demographics of training data and size of training data.

Skip Gram Negative Sampling (SG300): We trained 300 dimensional word embeddings separately using the SGNS (Mikolov et al., 2013) model. We restricted the training data to training set only for each experiment.

Glove: We used the publicly available Glove embeddings[1] (Pennington et al., 2014) trained on Wikipedia 2014 corpus with dimension sizes 50 (**G50W**) and 300 (**G300W**) and another 300 dimensional embeddings (**G300C**) trained on a significantly larger Common Crawl dataset.

Character Embeddings (CHAR): Character level features can be useful to handle rare words and spelling errors which are usually OOV for word embedding models. To use this character-level knowledge, we employ a convolutional neural network (CNN) with 30 filters of fixed window of size 3. We perform max pooling on output of convolution operations to generate 100 dimensional embeddings for each word. This character-level representation is then concatenated with the corresponding word embedding and fed into the network. Such character level embeddings have been shown to have potential to replace hand crafted character features. (Chiu and Nichols, 2015)

4.2 Context Encoder (CE)

The context encoder we implemented is a unidirectional LSTM network. At every time step tokens from the latest system utterance are fed as inputs to the context encoder. The encoder updates its internal state thus transforming the system utterance to a rich fixed sized representation which is then fed to the tagger's forward hidden state. The context encoder enables the system to scale across various domain and language settings by maintaining the immediate history.

4.3 Tagger

The Tagger network is responsible for performing the NER on user utterances.

Recurrent Neural Network (RNN): RNN (Elman, 1990; Ubeyli and ¨ Ubeyli, 2012) consists of a hidden state that depending on the previous hidden state and current input continually updates itself at every time step. The output is then predicted on the basis of the new hidden state.

Long Short Term Memory (LSTM): LSTMs (Hochreiter and Schmidhuber, 1997) a modification to RNNs introduce additional gated mechanisms to manage the vanishing/exploding gradient problems faced by RNN.

Bidirectional LSTM (BI-LSTM): In NER, at a given time step we have access to both past and future inputs. This gives us an opportunity to implement a BI-LSTM architecture.

4.4 Conditional Random Fields (CRF)

We add a linear chain CRF (Lafferty et al., 2001) network on top of the output states (concatenated forward and backward states at each time step) yielded by the tagger network to form BI-LSTM-CRF model. This layer considers dependencies across output labels to compute the log likelihood of IOB sequence tags using Viterbi decoding algorithm.

[1] https://nlp.stanford.edu/projects/glove/

Model	SGNS300	G50W	G300W	G300C
BI-LSTM	86.928	88.138	89.388	90.057
BI-LSTM-CE	89.130	90.163	90.910	91.224
BI-LSTM-CHAR	87.465	89.089	89.442	90.551
BI-LSTM-CHAR-CE	89.412	91.087	91.342	91.880
BI-LSTM-CRF	87.782	89.529	89.871	90.627
BI-LSTM-CRF-CE	89.696	91.122	91.455	92.133
BI-LSTM-CHAR-CRF	88.276	89.628	90.971	91.079
BI-LSTM-CHAR-CRF-CE	**90.036**	**91.705**	**92.042**	**92.864**

Table 1: Macro Averaged F1 scores on the DSTC-FRAMES-EN dataset

5 Experiments

We evaluated the performance of 3 major recurrent neural cell (RNN, GRU, LSTM) types on DSTC-FRAMES-EN by constructing unidirectional recurrent networks using these cells. The LSTM, GRU (Cho et al., 2014) based networks showed significant improvement over RNN architecture. The LSTM architecture displayed a marginal increase in performance in comparison to the GRU network. We thus conducted our further experiments by using the LSTM cell.

All BI-LSTM networks presented are stacked two layers deep with each cell containing 64 hidden units. We train our models with Adam (Kingma and Ba, 2014) optimizer for up to 30 epochs and use early stopping to avoid overfitting.

Since there are no pre-trained Glove embeddings for Hindi and transliterated Hindi, we restricted our set of experiments on DSTC-FRAMES-ENHI to only SGNS embeddings we trained separately

Choice of Architecture: As shown in Table 1 and Table 2, the character level embeddings helped increase the performance of the network by leveraging character level features and handling OOV tokens. With an addition of slightly more parameters the CRF layer boosted the system's performance. The networks which included the context encoder showed significant improvements in comparison to their non-context encoder counterparts.

Choice of word embeddings: From Table 1, we can see that the G50W displayed better performance than SG300 owing to a larger corpus of training data. The G300W being trained on the same training corpus as G50W exhibited improved results on account of larger dimension size. The best results were displayed by G300C which as trained on a larger and better suited style of data for our conversational settings.

Model	SGNS300
BI-LSTM	84.867
BI-LSTM-CE	86.242
BI-LSTM-CHAR	85.119
BI-LSTM-CHAR-CE	86.433
BI-LSTM-CRF	85.342
BI-LSTM-CRF-CE	86.790
BI-LSTM-CHAR-CRF	85.643
BI-LSTM-CHAR-CRF-CE	**87.934**

Table 2: Macro Averaged F1 scores on the DSTC-FRAMES-ENHI dataset

6 Conclusion and Future Work

We presented results for NER from variants of the popular BI-LSTM architecture in a task-oriented conversational setting. Adding a CRF layer boosts performance at slightly extra computational cost. Our results show that context in form of system utterance just before the user query potentially has important information about domain and slots and including it further boosts performance. Although we only included just one utterance from conversational history, including more conversation history can be helpful but can also be challenging as it might put pressure on the context encoder to ignore already detected slots. Nevertheless, it remains to be explored for future work. We also find that NER models also benefit from large pre-trained word representations and character level representations.

References

Yanqing Chen and Steven Skiena. 2016. False-friend detection and entity matching via unsupervised transliteration. *arXiv preprint arXiv:1611.06722*.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724– 1734. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10,* EMNLP '02, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Leon´ Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: a corpus for adding memory to goal-oriented dialogue systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 207– 219. Association for Computational Linguistics.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272. Association for Computational Linguistics.

Sepp Hochreiter and Jurgen¨ Schmidhuber. 1997. Long short-term memory. *Neural computation,* 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR,* abs/1412.6980.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv*:1508.01991.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML'01, pages282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270. Association for Computational Linguistics.

Dekang Lin and Xiaoyun Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume* 2, ACL '09, pages 1030–1038, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 879–888. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, Advances in *Neural Information Processing Systems 26,* pages 3111–3119. Curran Associates, Inc.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 78–86. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. *In Proceedings of the 2014 Conference on Empirical Methods in Natural Language*

Processing (EMNLP), pages 1532–1543. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227– 2237. Association for Computational Linguistics.

L. R. Rabiner, C. H. Lee, B. H. Juang, and J. G. Wilpon. 1989. Hmm clustering for connected word recognition. In *International Conference on Acoustics, Speech, and Signal Processing*,, pages 405–408 vol.1.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 147–155, Stroudsburg, PA, USA. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*.

Cicero dos Santos and Victor Guimaraes˜. 2015. Boosting named entity recognition with neural character embeddings. In *Proceedings of the Fifth Named Entity Workshop*, pages 25–33. Association for Computational Linguistics.

Elif Derya Ubeyli and Mustafa Ubeyli. 2012. Case studies for applications of elman recurrent neural networks.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Improving Computer Generated Dialog with Auxiliary Loss Functions and Custom Evaluation Metrics

Thomas Conley
University of Colorado
Colorado Springs
1420 Austin Bluffs Pkwy
Colorado Springs, CO, USA
tconley@uccs.edu

Jack St. Clair
Haverford College
370 Lancaster Ave
Haverford, PA, USA
jrstclair@haverford.edu

Jugal Kalita
University of Colorado
Colorado Springs
1420 Austin Bluffs Pkwy
Colorado Springs, CO, USA
jkalita@uccs.edu

Abstract

Although people have the ability to engage in vapid dialogue without effort, this may not be a uniquely human trait. Since the 1960's researchers have been trying to create agents that can generate artificial conversation. These programs are commonly known as chatbots. With increasing use of neural networks for dialog generation, some conclude that this goal has been achieved. This research joins the quest by creating a dialog generating Recurrent Neural Network (RNN) and by enhancing the ability of this network with auxiliary loss functions and a beam search. Our custom loss functions achieve better cohesion and coherence by including calculations of Maximum Mutual Information (MMI) and entropy. We demonstrate the effectiveness of this system by using a set of custom evaluation metrics inspired by an abundance of previous research and based on tried-and-true principles of Natural Language Processing.

Introduction

Computer scientists have tried to build chatbots for a long time, starting from the initial attempt at building an artificial psycho-therapist called Eliza (Weizenbaum, 1966). Because of the nature of psychotherapy, even with its limited abilities, Eliza was able to impress the populace at large, in addition to the research community. Eliza worked simply by pattern matching, and produced inane responses when pattern matching failed to produce a meaningful response.

The frame-based architecture used by (Bobrow et al., 1977) in the GUS system was the predominant approach to building dialog agents for several decades. Apple's SIRI and other digital assistants have used this architecture (Bellegarda, 2013, 2014; Jurafsky and Martin, 2018). Such speech-based conversation agents used a Partially Observable Markov Decision Process (Sondik, 1971) in a frame-based architecture, to maintain a system of beliefs and updated the system using Bayesian inference. They also used reinforcement learning (Sutton and Barto, 1998) as necessary.

Recently, researchers have started building chatbots by training machine learning programs on transcripts of conversations. Ritter, Cherry, and Dolan (2011) presented a data-driven approach to generating responses to Twitter status posts, using statistical machine translation, treating a status post as a question and the response as its "translation". Of late, researchers have built chatbots using Artificial Neural Networks (ANN) or Deep Learning (Cho et al., 2014; Sutskever, Vinyals, and Le, 2014). ANN-based Seq2Seq models have been used by many recent chatbots (Vinyals and Le, 2015; Li et al., 2016b,a; Shao et al., 2017; Wu, Martinez, and Klyen, 2018).

Although the Seq2Seq framework has shown good results in dialogue generation, we believe that the evaluation of the dialogues can be better measured. The research presented in this paper examines the role that various auxiliary loss functions play in the quality of generated dialog by RNNs when trained on several conversational corpora. Our contribution lies in the detailed analysis of generated dialogues, using custom metrics, as we change the auxiliary loss function. We believe that this is the first time such detailed analysis of automatically generated dialogs has been carried out. We use a simple RNN model for training the conversation agents since our primary focus is on auxiliary loss functions. We believe that this approach will have general applicability in other neural network architectures as well.

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 143–149

Problem Statement

We define a dialogue as the sequence of text elements $\mathcal{D}$ generated by the interaction between two agents $\mathcal{Q}$ and $\mathcal{A}$. Text elements are a sequence of characters, $t \in \{c_1, c_2, ..., c_i\}$, where c_i is a character from used in the words of the conversation vocabulary. Each elements t_i is shown as q_i or a_i to distinguish outputs from agents $\mathcal{Q}$ and $\mathcal{A}$ respectively. A conversation is seeded with an initial text element q_1, and $\mathcal{A}$ responds with a follow-up statement a_1. As shown in Equation 1,

$$\mathcal{D} = \langle\langle q_1, a_1\rangle, \langle q_2, a_2\rangle, \ldots, \langle q_i, a_i\rangle\rangle \quad (1)$$

the sequence grows with the continuous application of function $\mathcal{R}(t)$ as in Equations 2 and 3,

$$a_i = \mathcal{R}(q_i) \quad (2)$$

$$q_{i+1} = \mathcal{R}(a_i) \quad (3)$$

which show that each element of the conversation is generated from previous elements. The function $\mathcal{R}(t)$ is a forward pass through an RNN using sequence t_i as input and is followed by a beam search of the RNN output. We improve sequence generation and the function $\mathcal{R}(t)$ by incorporating auxiliary loss functions during the beam search.

A typical loss function in the context of classification, computes error by comparing predicted values with true values; the errors are propagated backward during training. However, a Seq2Seq model trains on a series of sequences without labeled answers, that is, without any knowledge of what the truth is. Instead, these models rely on minimizing the cross-entropy between the input and the raw network output. No output sequences are created during training.

We present auxiliary loss functions which are applied after training during sequence generation by the beam search. Each path through the answer space represents a single possible choice for the final sequence. The best answer among all possible paths is chosen by optimization of these loss function.

Finally, we present simple evaluation metrics for determining the efficacy of our dialogue generation model. ‘

Related Work

Using Seq2Seq models for dialogue generation has become commonplace in recent years. Ritter, Cherry, and Dolan (2011) were the first to use a model used for Statistical Machine Translation (SMT) to generate responses to queries by training on a corpus of query-response pairs. Sordoni et al. (2015) improved Ritter et al.’s work by re-scoring the output of the SMT-based response generation system with a Seq2Seq model that took context into account.

Vinyals and Le (2015) used an RNN-based model with a cross-entropy based auxiliary loss function and a greedy search at the output end. Wen et al. (2015) used LSTMs for joint planning of sentences and surface realization by adding an extra cell to the standard LSTM architecture (Hochreiter and Schmidhuber, 1997), and using the cross-entropy loss. They produced sentence variations by sampling from sentence candidates. Li et al. (2016a) used Maximum Mutual Information (MMI) as the objective function to produce diverse, interesting and appropriate responses. This objective function was not used in the training of the network, but to find the best among candidates produced by the model at the output, during generation of responses. Our paper is substantially inspired by this work.

Li et al. (2016b) applied deep reinforcement learning using policy gradient methods to punish sequences that displayed certain unwanted properties of conversation: lack of informativity, incoherence and responding inanely. Lack of informativity was measured in terms of high semantic similarity between consecutive turns of the same agent. Semantic coherence was measured in terms of mutual information, and low values were used to penalize ungrammatical or incoherent responses.

Su et al. (2018) use a hierarchical multi-layered decoding network to generate complex sentences. The layers are GRU-based (Cho et al., 2014), and each layer generates words associated with a specific Part-Of-Speech (POS) set. In particular, the first layer of the decoder generates nouns and pronouns; the second layer generates verbs, the third layer adjectives and adverbs; and the fourth layer, words belonging to other POSes. They also use a technique called teacher forcing (Williams and Zipser, 1989) to train RNNs using the output from the prior step as an input.

Despite the relatively new methods that are being proposed for question answering and dialogue generation, the evaluation of the the generated text still relies on metrics like BLEU (Bilingual Evaluation Understudy) (Papineni et al., 2002), a metric that was designed for evaluation of SMT. BLEU computes scores for individual translated

sentences by comparing overlaps in terms of n-grams with a set of good quality reference translations. These measurments alone are insufficient for evaluating the effectiveness of dialogue generation systems.

Li et al. (2016b) used two additional computable metrics: the length of the dialogue generated, and diversity of distinct unigrams and bigrams. While this simple measure may be a good addition to BLEU we, believe that a wider set of evaluation metrics is needed. Coh-Metrix (Graesser et al., 2004) is a Web-based tool that analyzes texts with over 100 measures of cohesion, language complexity, and readability. We have used Coh-Metrix extensively in the evaluation of dialogue from this research and it has provided a rich understanding of the quality of our results.

Loss Functions

Our training model employs a softmax cross entropy loss function for back-propagation during training. Rather than modify this primary loss function, we concentrate on the auxiliary loss function needed during sentence generation. This function operates on partially generated sequences during a beam search and is used to find consensus among a number of possible choices equal to the beam width. We have tested extensively using a beam width of 2 since our functions are configured to process 2 parameters. We leave the expansion of this process to handle wider beam widths as an obvious future enhancement.

We begin our testing using no auxiliary loss function at all and rely on network predictions alone to select subsequent characters. We call this Network Loss (NET) in this research and consider the results a control baseline for comparison with other functions.

We continue testing with a basic MMI loss function $\hat{T}_{MMI}$ as shown in Equation 4, where S represents the current set of solution states during sentence generation and T represents the set of possible next states. This function is modeled after work conducted by (Li et al., 2016a). The weighting factor λ is configurable at run time and is used to adjust the relevance of current solution states versus future solution states, in the decision process.

$$\hat{T}_{MMI} = \arg\max_{T} \{ \log p(T|S) - \lambda \log p(T) \} \quad (4)$$

The basic MMI approach is suggested by (Estévez et al., 2009) and implemented as shown in Equation 4. We further develop this MMI approach by including Entropy normalization, as inspired by (Trinh et al., 2018) by who used normalized MMI for feature selection. We calculate entropy from predicted network probabilities as shown in Equations 5 and 6.

$$H_S = \sum_{t=0}^{|S|} -P(S_t) \times log(P(S_t)) \quad (5)$$

$$H_T = \sum_{t=0}^{|T|} -P(T_t) \times log((T_t) \quad (6)$$

The minimum of these values is used to normalize our MMI value as in Equation 7.

$$\hat{T}_{NORM} = \frac{\hat{T}_{MMI}}{min(H_S, H_T)} \quad (7)$$

Finally we experiment with MMI entropy normalization where entropy is not calculated but measured directly from the training corpus in terms of character frequencies. Optimizing based on this function should affect the uniqueness of generated sentences.

Architecture

The core of our model is a stack of dense layers comprised of gated recurrent unit (GRUs) cells. We tested extensively on a configuration with 3 layers, each divided into 3 blocks, where each block contained 2048 GRUs. This architecture is based on a prior implementation available online[1].

The GRU stack is initialized with the previous state (s_{t-1}) and the current character encoding (x_t) at each time step t in the character sequence. The GRU output (Y_t) and the weights from the final stack layer (W_t) are combined with a bias (b) to produce logits at time t. We define logits as the raw output of the GRU stack which can be normalized and passed to a softmax function to produce probabilities. In this scheme, we update the logits by applying weights and biases from the last GRU layer as shown in Equation 8.

$$Logits = (Output \times Weights) + Biases \quad (8)$$

The logits are then passed to a loss function for back propagation within the GRU stack. We do not limit or pad the length of the input sequence but

[1] https://github.com/pender/chatbot-rnn

perform back propagation through time (BBTT), relying on TensorFlow's default truncated back-propagation capabilities. Note that, output sequences $(y_0, ..., y_t)$ are not generated during the training phase where only the logits are used for back-propagation. It is after training, during testing or dialogue generation, that the logits are converted to probability using softmax. Finally, probabilities are converted to character sequences using a beam search.

Our beam search employs custom loss functions based on Maximum Mutual Information (MMI) as described in (Li et al., 2016b). We extend this concept to include entropy-normalized MMI as discussed previously. Figure 1 illustrates a single time-step t in sequence processing by our recurrent neural network.

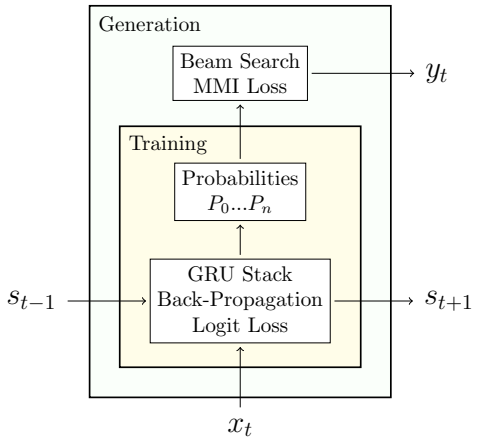


Figure 1: Custom Loss Model

The model accepts a (one-hot) binary vector X and a previous state vector, S, as inputs and produces a state vector, S and a predicted probability distribution vector P_t, for the (one-hot) binary vector Y_t.

Evaluation Metrics

Evaluation of generated text remains a difficult task as there is little consensus regarding what makes a good conversation (Liu et al., 2016). Word-overlap metrics such as BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005) and ROUGE (Lin, 2004) have been used in the past, however, a simple overlapping of words between question and answer may not make for a good conversation and repetition may be considered annoying and reminiscent of Eliza, as mentioned earlier.

We begin our testing of generated dialogue using the on-line suite of tools provided by CohMetrix (Graesser et al., 2004). Although this is a very manual process of cutting and pasting results, it provides insight from over 100 different metrics related to cohesion and coherence of text.

After examining several of these measurements for effectiveness in evaluating our dialogues; we use the knowledge gained from this manual process to develop a few simple metrics that reflect the concepts of cohesion and coherence, but can be automated. We built our simple metrics using tried-and-true NLP standard modules such as WordNet (Fellbaum, 1998), GloVe (Pennington, Socher, and Manning, 2014), NLTK (Loper and Bird, 2002) and the Stanford CoreNLP (Manning et al., 2014).

Inspired by the fore mentioned tools, we present four simple distance functions which we apply to sentences pairs from generated dialogues as a measure of coherence and cohesion.

- *SynSet Distance* This metric uses a human generated semantic knowledge-base (WordNet) to create two sets of semantic elements, where elements consist of synonyms and lemmas evoked by the words of each sentence. The ratio of the intersection of the sets to the union of the sets provides a distance measurement between 0 and 1.

- *Embedding Distance* Here we exploit the semantic knowledge inherent in pretrained word embeddings to produce a set of the n-closest words from each word in a sentence. Similar to *SynSet Distance* we use the ratio of the intersection of the sets to the union of the sets get a value between 0 and 1.

- *Cosine Distance* We consider that the set of word embeddings from a sentence has semantic meaning in a manner similar to the well known concept of "bag-of-words". The cosine distance between the average of the two sets provides a result between 0 and 1.

- *Sentiment Distance* A Naive Bayes Analyzer provides a simple measure of positive or negative sentiment, for each sentence. With values between 0 and 1, a simple difference is used to represent *Sentiment Distance*.

Experiments and Results

We tested our model by training on dialogue from Reddit and from the proceedings of the Supreme Court of the United States (SCOTUS) and by using four distinct auxiliary loss functions described in this research. Network loss (NET), Maximum Mutual Information (MMI), Normalized MMI (NORM) and Entropy Normalized MMI (ENT) were used to generated conversations consisting of 15 question and answer pairs for testing.

Using the Reddit trained model, multiple tests were run using Coh-Metrix and some results are summarized in Table 1. All test conversations consist of 15 question and answer pairs generated by two different chatbots. This summary of results provides insight into the relative effectiveness of our loss models as measured by Coh-Metrix. The definition of these metrics is left to (Graesser et al., 2004); however observed trends in Coh-Metrix have led to the development of our own custom metrics.

	NET	MMI	NORM	ENT
Mean Words per Sentence	10.070	3.200	1.550	51.389
Narrativity	99.910	98.170	57.140	78.810
Syntactic Simplicity	58.320	41.680	99.930	0.160
Referential Cohesion	90.820	64.800	100	100
Sentence Semantic Similarity	0.363	0.359	0.167	0.624
Lexical Diversity	0.366	0.594	0.333	0.096
Connective Word Occurrence	48.499	0	0	57.297
Modifiers per Noun Phrase	0.408	0.231	0	0.908
Sentence Syntax Similarity	0.114	0.158	0.593	0.040
Content Word Frequency	2.813	4.580	2.358	2.835
Word Familiarity	589.115	572	591.5	583.183
Reading Ease	90.526	100	98.835	63.476

Table 1: Selected Coh-Metrix results from our model using four auxiliary loss functions.

Comparative results shown in Figure 2 indicate lower values for all 4 non-random metrics, showing that our system is not just parroting text sequences from the training corpus. The larger results, produced by ENT, indicate that entropy normalization increases uniqueness in responses and thus increases the distance measure, as expected. The lower measurements for the MMI based functions indicate a closer cohesion and coherence between question and answer; this may be a result of using lambda factor equal to .5 during testing which reduces the impact of previous solution states in favor of the predicted solution state.

Cohesion and Coherence

A generated sample of text from SCOTUS, shown in Table 2 illustrates the difference between cohesion and coherence. The fact that sentences seem

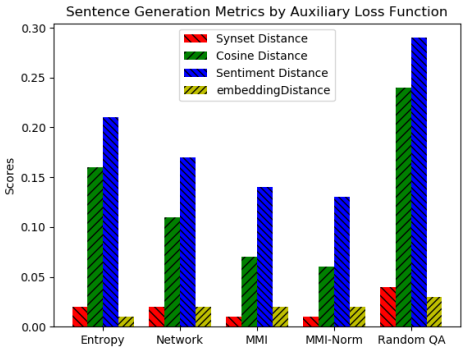


Figure 2: Average distance measurements for custom auxiliary loss functions across all datasets.

to fit together well and flow naturally indicate high cohesion which can be produced by the neural network alone. However, a close reading of the text shows that the network was unable to give logical sense to the words and sentences. The capitalization at the beginning of the sequence correctly shows name of a random speaker, as found in the training corpus. Our testing shows that a network built on a larger training set has greater cohesion dialogue of Table 2 is reasonable, but no level of training alone was able to create dialogue with any real logic or meaning.

”MR. COLE: I think we’re talking about the district court to review it does, Your Honor. I believe that’s correct, Justice Ginsburg. It’s – it’s in navigation. If you have the distinction between aliens who we collect taxes. They’re – they’re contested, would be able to read the restatement of the landowners – or – or that decision. In that instance, I think that was referred to the issue before this Court that have standing alone and then have set forth in these kinds of prosecutions, when i”

Table 2: Generated response from SCOTUS showing reasonable cohesion but a lack of coherence.

Conclusion and Future Work

Advancements in technology may allow development of more complex neural networks and more sophisticated loss functions. With better evaluation models, a neural-network-based chatbot may be enhanced to learn more from itself using a better form of back-propagation, during the generation phase, as described in this research.

Although human interaction is still considered to be the best method for dialog evaluation, future dialog generation models, based on this research, may be able to bring human level sophistication to computer generated text.

References

Banerjee, S., and Lavie, A. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, 65–72.

Bellegarda, J. R. 2013. Natural language technology in mobile devices: Two grounding frameworks. In *Mobile Speech and Advanced Natural Language Solutions*. Springer. 185–196.

Bellegarda, J. R. 2014. Spoken language understanding for natural interaction: The siri experience. In *Natural Interaction with Robots, Knowbots and Smartphones*. Springer. 3–14.

Bobrow, D. G.; Kaplan, R. M.; Kay, M.; Norman, D. A.; Thompson, H.; and Winograd, T. 1977. Gus, a frame-driven dialog system. *Artificial intelligence* 8(2):155–173.

Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734.

Estévez, P. A.; Tesmer, M.; Perez, C. A.; and Zurada, J. M. 2009. Normalized mutual information feature selection. *IEEE Transactions on Neural Networks* 20(2):189–201.

Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Graesser, A. C.; McNamara, D. S.; Louwerse, M. M.; and Cai, Z. 2004. Coh-metrix: Analysis of text on cohesion and language. *Behavior research methods, instruments, & computers* 36(2):193–202.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Jurafsky, D., and Martin, J. 2018. *Speech & Language Processing (Third edition draft, available at https://web.stanford.edu/ jurafsky/slp3*. Pearson.

Li, J.; Galley, M.; Brockett, C.; Gao, J.; and Dolan, B. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 110–119.

Li, J.; Monroe, W.; Ritter, A.; Galley, M.; Gao, J.; and Jurafsky, D. 2016b. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Liu, C.-W.; Lowe, R.; Serban, I.; Noseworthy, M.; Charlin, L.; and Pineau, J. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2122–2132.

Loper, E., and Bird, S. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, 63–70. Stroudsburg, PA, USA: Association for Computational Linguistics.

Manning, C. D.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S. J.; and McClosky, D. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, 55–60.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, 311–318. Association for Computational Linguistics.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543.

Ritter, A.; Cherry, C.; and Dolan, W. B. 2011. Data-driven response generation in social media. In *Proceedings of the conference on empirical methods in natural language processing*, 583–593. Association for Computational Linguistics.

Shao, Y.; Gouws, S.; Britz, D.; Goldie, A.; Strope, B.; and Kurzweil, R. 2017. Generating high-quality and informative conversation responses with sequence-to-sequence models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2210–2219.

Sondik, E. J. 1971. The optimal control of partially observable markov decision processes. *PhD the sis, Stanford University*.

Sordoni, A.; Galley, M.; Auli, M.; Brockett, C.; Ji, Y.; Mitchell, M.; Nie, J.-Y.; Gao, J.; and Dolan, B. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*.

Su, S.-Y.; Lo, K.-L.; Yeh, Y. T.; and Chen, Y.-N. 2018. Natural language generation by hierarchical decoding with linguistic patterns. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, 61–66.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, 3104–3112.

Sutton, R. S., and Barto, A. G. 1998. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Trinh, T. H.; Dai, A. M.; Luong, T.; and Le, Q. V. 2018. Learning longer-term dependencies in rnns with auxiliary losses. *CoRR* abs/1803.00144.

Vinyals, O., and Le, Q. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Weizenbaum, J. 1966. ELIZA – a computer program for the study of natural language communication between man and machine. *Communications of the ACM* 9(1):36–45.

Wen, T.-H.; Gasic, M.; Mrksic, N.; Su, P.-H.; Vandyke, D.; and Young, S. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Williams, R. J., and Zipser, D. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1(2):270–280.

Wu, X.; Martinez, A.; and Klyen, M. 2018. Dialog generation using multi-turn reasoning neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, 2049–2059.

Analyzing Autism Speech of Children in English Vowels Regions by Analysis of Changes in Production Features

Abhijit Mohanta[1] **and Vinay Kumar Mittal**[2]
[1]Indian Institute of Information Technology, Sri City, Chittoor, Andhra Pradesh, India
[2]CEO, Ritwik Software Technologies Pvt. Ltd, Hyderabad, Telangana, India
abhijit.mohanta@iiits.in[1] and DrVinayKrMittal@gmail.com[2]

Abstract

Children with autism spectrum disorder (ASD) have difficulty in producing the speech that is different from the speech of normal children. Most children with ASD have difficulty in proper communication of information, their thoughts or their emotional state. Only a few studies have been carried out towards acoustic analysis of ASD speech. Objective of this study is to characterize the speech signal of the children affected with autism, by examining changes in the acoustic features. An autism speech dataset has been collected from the children affected with autism, for over a year. Autism speech is examined mainly in English vowels regions due to their relatively longer duration, and quasi-periodic nature of the vocal folds during pronunciation of the vowel sounds. Changes in the characteristics of autism speech are analyzed by examining changes in the production features. The excitation source characteristics are examined using the feature F0, and the vocal tract filter, i.e., system characteristics, by using dominant frequencies features. The combined characteristics of the source-system interaction are examined using features SoE, ZCR and signal energy. Changes are examined in five English vowels regions. Distinct patterns of changes observed in the autism speech of male and female children are discussed.

keywords: ASD, ZFF, F0, FD1, FD2

1 Introduction

Autism spectrum disorder (ASD) is a pervasive developmental disorder, defined clinically by observing abnormalities in three areas: communication, social reciprocity, and reduced or hyperfocus behavioral flexibility (Chaspari et al., 2014; Kjelgaard and Tager-Flusberg, 2001). It is knows as a spectrum disorder because of its heterogeneity of symptomatology (Bone et al., 2012). According to (Black et al., 2011; Mower et al., 2011a), 1 in 110 children are with ASD. Disturbances of prosody, communication impairments and abnormalities involving speech impairments are some of the most common aspects among many individuals with ASD (Fusaroli et al., 2017). Also, individuals with autism carry some specific biomarkers connected with the disorder by birth, and these biomarkers develop in such a way that can be clearly identified by 36 months of life or later (McCann and Peppé, 2003). However, there are no fixed rules to define autism, also the biological and genetic reasons behind the ASD are still unknown (Herbert et al., 2005). Still, only a few researches have been done on *autism speech*, i.e., the speech of ASD affected children. The purpose of this research is to analyze speech signal of the children with autism in English vowels regions, by examining changes in the production features.

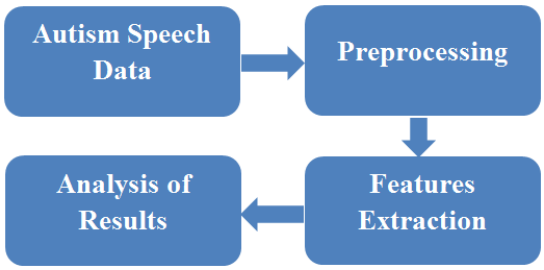


Figure 1: Block diagram of the proposed plan.

Sometimes the individuals with autism are associated with the difficulties in expressing their emotions as well as understanding others' emotional states from speech, facial expression, etc (Marchi et al., 2012). The reason behind language

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 150–154

Table 1: Datset details of the children with ASD.

Attributes	Statistics
Total number of the children	13 (11 male, 2 female)
Age (in years)	3.5 to 16
Native language	Tamil: 12 and Punjabi: 1
Reading skill (English)	Beginner: 2 Intermediate: 4 Fluent: 7
Total files	187
Duration of data	9350 sec

impairment in autism is the outcome of primary linguistic disorder with a focus on pragmatic impairments (Rutter, 1970; Baltaxe, 1977). In fact, in many cases, a significant spoken language delay and repetitive language could be encountered in the children with ASD (Mower et al., 2011b). In general, the children with typical development start establishing their vocabularies at the age of 24 months, but the children with autism could be unable to do the same (Short and Schopler, 1988). Also, compared to the individuals with typical speech, individuals with high-functioning autism (HFA) have a large variation in pitch, and some of them have the absence of terminal pitch contour in their speech (Shriberg et al., 2001; Diehl et al., 2009).

This paper discusses the analyzation of autism speech in the vowel regions of five English vowels. Speech dataset collection is one of the most challenging tasks in order to do the research on speech signal of the children with autism. Although, for this research purpose an English speech signal dataset has been collected by recording speech samples of the children with ASD. Only the vowels regions of English language have been considered in this study because of their relatively longer duration and the presence of sustained speech signal in vowel regions. The excitation source characteristics are examined using the feature instantaneous fundamental frequency (F0), and the system characteristics are examined using the first two dominant frequencies (FD1 and FD2). In addition, the combined characteristics are examined using the features strength of excitation (SoE), zero crossing rate (ZCR) and signal energy (E). All these features have been extracted only form the vowels parts of the speech signal.

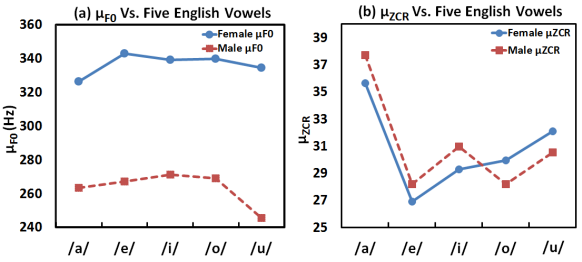


Figure 2: Average F0 and ZCR for male and female children with autism: (a) μ_{F0}, (b) μ_{ZCR}.

This study consists of four major steps, graphically represented in Figure 1. Firstly, a speech signal dataset was collected, by recording the sound files of the children with ASD. Secondly, in preprocessing step, unwanted signal parts were removed, and the speech signal files were arranged in a database. Thirdly, speech signal processing methods were applied on the collected dataset to extract the production features. Lastly, results were made by observing changes in the extracted features.

The organization of this paper is as follows. Details about the database collection of the children with ASD are discussed in section 2. Next, the signal processing methods and features used for the purpose of analysis are discussed in section 3. Section 4 presents the key results, observations and discussion on results. Lastly, section 5 presents the conclusions along with the scope of future work on this topic.

2 Speech Signal Dataset of ASD Children

A speech signal dataset of the children with ASD was recorded for this research purpose. Dataset details are tabulated in Table 1. Here, in this study, the children with age less than 41 months were not considered, because according to a study ASD start in the first 36 months of life (McCann and Peppé, 2003). Dataset was recorded in English. All the speakers who were selected for data collection had a knowledge of English, and also had some speaking related problems. It was made sure by a certified doctor that the children considered for the data collection were diagnosed with ASD. Also, before data collection it was made sure that they met the DSM–IV diagnostic criteria (Lord et al., 1994) and other diagnostic criteria for autism.

Data was collected once or twice in a week for a year period of time in a noise-free empty room.

Table 2: Mean (μ) values of the acoustic features of the male children with autism: (a) acoustic features and (b)-(f) mean values in five English vowels regions.

(a) Features	(b) /a/	(c) /e/	(d) /i/	(e) /o/	(f) /u/
F0 (Hz)	263.3	267.1	271.3	269.1	254.7
E×1000	36.53	31.41	43.18	41.25	54.29
SoE×100	34.9	43.4	44	39.1	35.9
ZCR×1000	37.7	28.18	30.98	28.21	30.55
FD1 (Hz)	1041.7	900.5	1043.4	863	951.8
FD2 (Hz)	3294.6	3234.2	3281.5	3291.4	3316.1

Table 3: Mean (μ) values of the acoustic features of the female children with autism: (a) acoustic features and (b)-(f) mean values in five English vowels regions.

(a) Features	(b) /a/	(c) /e/	(d) /i/	(e) /o/	(f) /u/
F0 (Hz)	326.4	343	339.3	339.9	334.6
E×1000	39.5	35.3	48.35	63.75	58.65
SoE×100	32	37.1	38.1	42.9	37.1
ZCR×1000	35.65	26.9	29.3	29.95	32.1
FD1 (Hz)	864.7	686.1	783.3	802.8	859.9
FD2 (Hz)	3185	3285.9	3268.8	3057.6	3112.3

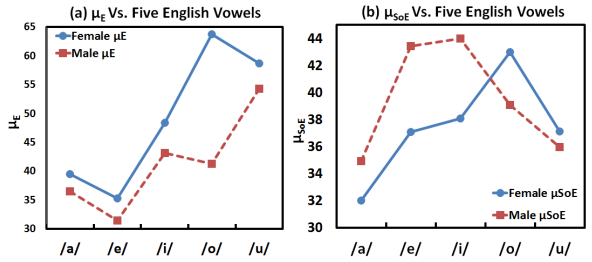


Figure 3: Average E and SoE for male and female children with autism: (a) μ_E, (b) μ_{SoE}.

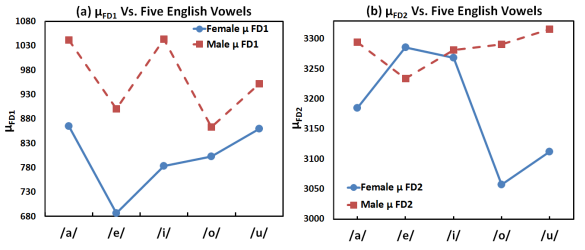


Figure 4: Average FD1 and FD2 for male and female children with autism: (a) μ_{FD1}, (b) μ_{FD2}.

The children were asked to pronounce a set of twenty five especially selected consonant-vowel-consonant (CVC) and consonant-vowel-vowel-consonant (CVVC) English words and numbers. These words and numbers were shown to them with pictures and text on a laptop. A study shows that children with autism have an early interest in letters and numbers, and that is a big reason behind showing them words and numbers (Volkmar et al., 1997; Tager-Flusberg et al., 2005).

Each child with ASD was asked to pronounce the same set of words, over the entire data collection period. There were five English words selected for each of the five English vowels. So, in a single day by each speaker, total utterances of 25 words were recorded for each of two such sections. Roland R-26 audio recorder was used for the recording purpose. In addition, data was recorded at a sampling rate of 48 KHz and in .wav format.

3 Signal Processing Methods and Features

The speech signal files of the children with ASD were analyzed by observing the changes in the source characteristics (F0), system characteristics (dominant frequencies), and combined characteristics (SoE, ZCR and signal energy). The F0 was derived using zero-frequency filtering (ZFF) method with the sampling frequency 10 KHz (Murty and Yegnanarayana, 2008; Yegnanarayana and Murty, 2009). The ZFF method involves computing the output of the cascade of two zero-frequency resonators (ZFRs) (Murty and Yegnanarayana, 2008; Yegnanarayana and Murty, 2009). The zero-frequency filter signal output of ZFR is given as:

$$y_1[n] = -\sum_{k=1}^{2} a_k y_1[n-k] + x[n] \qquad (1)$$

where, x[n] is pre-processed input signal, $a_1 = -2, and\ a_2 = 1$. This operation is repeated twice, for a cascade of ZFRs. The trend in this output is removed by subtracting the moving average corresponding to the 10 ms window at each sample. The resultant trend removed signal, called ZFF signal is given by:

$$y[n] = y_2[n] - \frac{1}{2N+1} \sum_{m=-N}^{N} y_2[n+m] \qquad (2)$$

where, 2N+1 is the window length in terms of sample number. The resultant signal is called the ZFF signal. Its positive giving zero crossings indicate the glottal closure instants (GCIs), which are used to estimate the F0 (Murty and Yegnanarayana, 2008). In addition, the slope of the ZFF signal around the GCIs gives a measure of the SoE (Murty and Yegnanarayana, 2008; Mittal and Yegnanarayana, 2015a).

The FD1 and FD2 were derived using linear prediction (LP) analysis (Makhoul, 1975). With the LP order 5, the LP spectrum will have a maximum of two peaks. The frequencies corresponding to these peaks are called the dominant frequencies, denoted as FD1 and FD2, respectively (Mittal and Yegnanarayana, 2015b).

The signal energy (E) (Rihaczek, 1968) was calculated using the frame size 30 ms and frame shift 10 ms. Signal energy of a discrete-time signal $x[n]$ can be computed as: $E_w = \sum_{n=-w/2}^{w/2} |x[n]|^2$ where, w is the window length.

4 Results and Observations

The μ_{F0}, μ_E, μ_{SoE}, μ_{ZCR}, μ_{FD1}, and μ_{FD2} values for male and female children with autism are represented in Table 2 and 3, respectively. From Figure 2(a), it is observed that for each of the five English vowels, the female children with autism have the higher vocal fold vibration rate than the male children with autism. Although, this statement is also true in case of the individual with typical speech. Also, in case of both the male and female children with autism, front vowels i.e., /e/ and /i/ give the highest μ_{F0} values as compared to other English vowels. Lastly, in case of the μ_{F0} values of the rear vowels i.e., /o/ and /u/, a similar pattern is observed for both the male and female children with autism.

The μ_{ZCR} values of the male and female speakers are represented graphically in Figure 2(b). In addition, μ_{ZCR} values are multiplied by 1000 for better understanding purpose. The μ_{ZCR} values of the front and mid vowels follow a similar trend for both the male and female speakers, which could be observed from Figure 2(b). Next, in case of front and mid vowels, the male speakers have the higher μ_{ZCR} values as compared to female, but in case of rear vowels the female speakers have the higher μ_{ZCR} values as compared to male speakers.

The μ_E values of male and female speakers are represented graphically in Figure 3(a). In addition, μ_E values are also multiplied by 1000 for better understanding purpose. Here, it is observed that the μ_E values of front and mid vowels for both the male and female speakers follow a similar trend. It could be observed from Figure 3(a).

All the μ_{SoE} values are multiplied by 100 for the purpose of better understanding. It is observed that in case of the front and mid vowels, μ_{SoE} values follow a similar trend for both the male and female speakers. Also, in case of the front and mid vowels, μ_{SoE} have the higher values for the male speakers as compared to female speakers. These statements could be observed from Figure 3(b).

The μ_{FD1} and μ_{FD2} values are represented in Figure 4(a) and 4(b), respectively. In case of all English vowels, μ_{FD1} values are higher for male children as compared to female children. Also, in case of female vowel, /e/ gives the lowest μ_{FD1} value and the highest μ_{FD2} value as compared to other English vowels. Next, in case of the female children, μ_{FD2} value of vowel /e/ is the highest as compared to other vowels. On the other hand, in case of the male children, μ_{FD2} value of vowel /e/ is the lowest as compared to other vowels.

5 Conclusions

The aim of this research is to analyze changes in the various speech production features in English vowel regions of children with ASD. An autism speech dataset is recorded for this research purpose. Changes are analyzed by observing the differences in the source characteristics (F0), system characteristics (FD1 and FD2), and combined characteristics (SoE, ZCR and E). In the conclusion, it could be stated that in case of the male and female children with ASD, front and mid vowels show the similar trend for F0, E, SoE and ZCR. But, in case of the rear vowels such trends are not present. These robust results could be used to differentiate between the children with autism and the typically developed individuals.

A small size of speech data for female children with ASD is a limitation of this study. More acoustic features could be considered for future studies.

Acknowledgments

The authors are thankful to the Doctrine Oriented Art of Symbiotic Treatment (DOAST) Integrated Therapy Centre for Autism, Anna Nagar, Chennai, India, for giving the opportunity to collect autism speech data.

References

Christiane AM Baltaxe. 1977. Pragmatic deficits in the language of autistic adolescents. *Journal of Pediatric Psychology*, 2(4):176–180.

Matthew P Black, Daniel Bone, Marian E Williams, Phillip Gorrindo, Pat Levitt, and Shrikanth Narayanan. 2011. The usc care corpus: Child-psychologist interactions of children with autism spectrum disorders. In *Twelfth Annual Conference of the International Speech Communication Association*.

Daniel Bone, Matthew P Black, Chi-Chun Lee, Marian E Williams, Pat Levitt, Sungbok Lee, and Shrikanth Narayanan. 2012. Spontaneous-speech acoustic-prosodic features of children with autism and the interacting psychologist. In *Thirteenth Annual Conference of the International Speech Communication Association*.

Theodora Chaspari, Matthew Goodwin, Oliver Wilder-Smith, Amanda Gulsrud, Charlotte A Mucchetti, Connie Kasari, and Shrikanth Narayanan. 2014. A non-homogeneous poisson process model of skin conductance responses integrated with observed regulatory behaviors for autism intervention. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 1611–1615. IEEE.

Joshua J Diehl, Duane Watson, Loisa Bennetto, Joyce McDonough, and Christine Gunlogson. 2009. An acoustic analysis of prosody in high-functioning autism. *Applied Psycholinguistics*, 30(3):385–404.

Riccardo Fusaroli, Anna Lambrechts, Dan Bang, Dermot M Bowler, and Sebastian B Gaigg. 2017. Is voice a marker for autism spectrum disorder? a systematic review and meta-analysis. *Autism Research*, 10(3):384–407.

Martha R Herbert et al. 2005. Autism: a brain disorder or a disorder that affects the brain. *Clinical Neuropsychiatry*, 2(6):354–379.

Margaret M Kjelgaard and Helen Tager-Flusberg. 2001. An investigation of language impairment in autism: Implications for genetic subgroups. *Language and cognitive processes*, 16(2-3):287–308.

Catherine Lord, Michael Rutter, and Ann Le Couteur. 1994. Autism diagnostic interview-revised: a revised version of a diagnostic interview for caregivers of individuals with possible pervasive developmental disorders. *Journal of autism and developmental disorders*, 24(5):659–685.

John Makhoul. 1975. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580.

Erik Marchi, Björn W Schuller, Anton Batliner, Shimrit Fridenzon, Shahar Tal, and Ofer Golan. 2012. Emotion in the speech of children with autism spectrum conditions: prosody and everything else. In *WOCCI*, pages 17–24.

Joanne McCann and Sue Peppé. 2003. Prosody in autism spectrum disorders: a critical review. *International Journal of Language & Communication Disorders*, 38(4):325–350.

Vinay Kumar Mittal and B Yegnanarayana. 2015a. Study of characteristics of aperiodicity in noh voices. *The Journal of the Acoustical Society of America*, 137(6):3411–3421.

Vinay Kumar Mittal and Bayya Yegnanarayana. 2015b. Analysis of production characteristics of laughter. *Computer Speech & Language*, 30(1):99–115.

Emily Mower, Matthew P Black, Elisa Flores, Marian Williams, and Shrikanth Narayanan. 2011a. Rachel: Design of an emotionally targeted interactive agent for children with autism. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pages 1–6. IEEE.

Emily Mower, Chi-Chun Lee, James Gibson, Theodora Chaspari, Marian E Williams, and Shrikanth Narayanan. 2011b. Analyzing the nature of eca interactions in children with autism. In *Twelfth Annual Conference of the International Speech Communication Association*.

K Sri Rama Murty and B Yegnanarayana. 2008. Epoch extraction from speech signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(8):1602–1613.

A Rihaczek. 1968. Signal energy distribution in time and frequency. *IEEE Transactions on information Theory*, 14(3):369–374.

Michael Rutter. 1970. Autistic children: infancy to adulthood. In *Seminars in psychiatry*, volume 2, page 435.

Andrew B Short and Eric Schopler. 1988. Factors relating to age of onset in autism. *Journal of autism and developmental disorders*, 18(2):207–216.

Lawrence D Shriberg, Rhea Paul, Jane L McSweeny, Ami Klin, Donald J Cohen, and Fred R Volkmar. 2001. Speech and prosody characteristics of adolescents and adults with high-functioning autism and asperger syndrome. *Journal of Speech, Language, and Hearing Research*, 44(5):1097–1115.

Helen Tager-Flusberg, Rhea Paul, Catherine Lord, F Volkmar, Rhea Paul, and Ami Klin. 2005. Language and communication in autism. *Handbook of autism and pervasive developmental disorders*, 1:335–364.

Fred R Volkmar, Kathy Koenig, and Matthew State. 1997. Childhood disintegrative disorder. *Handbook of Autism and Pervasive Developmental Disorders, Volume 1, Third Edition*, pages 70–87.

B Yegnanarayana and K Sri Rama Murty. 2009. Event-based instantaneous fundamental frequency estimation from speech signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(4):614–624.

Hate Speech Detection from Code-mixed Hindi-English Tweets Using Deep Learning Models

Satyajit Kamble
K J Somaiya College of Engineering
Mumbai, India
satyajit.k@somaiya.edu

Aditya Joshi
CSIRO Data61
Sydney, Australia
aditya.joshi@csiro.au

Abstract

This paper reports an increment to the state-of-the-art in hate speech detection for English-Hindi code-mixed tweets. We compare three typical deep learning models using domain-specific embeddings. On experimenting with a benchmark dataset of English-Hindi code-mixed tweets, we observe that using domain-specific embeddings results in an improved representation of target groups. We also show that our models result in an improvement of about 12% in F-score over a past work that used statistical classifiers.

1 Introduction

Hindi is one of the official languages of India[1], spoken by more than 551 million speakers[2]. As is typical of social media in any language, Hindi speakers on social media occasionally manifest hate towards one another. Hate speech refers to the use of hateful language, tone or prosody directed towards a person or a group of individuals, with the negative intention to provoke, intimidate, express contempt or cause harm to them. The membership to a group could be based on attributes such as race, religion, sexual orientation, ethnic origin, disability and so on.

Hate speech detection is the automated task of detecting if a piece of text contains hate speech. Hateful messages can be used to misinform people or result in violent incidents arising due to hate, therefore, hate speech detection assumes importance. In a recent news report, the Indian Government also expressed its intention to introduce a law to deal with online hate speech[3]. A tool for hate speech detection on social media in India is the need of the day.

As a country with high internet penetration and rich linguistic diversity, hate speech detection assumes an additional change in the case of Indian languages (Bali et al., 2014). Due to the difficulties in typing tools and familiarity with the English QWERTY keyboard, using a mixture of English words and transliterated Indian language words is common amongst the Indian internet users. Referred to as code-mixing or code-switching, the phenomenon corresponds to the use of transliterated words from one or more languages along with words in the language of the script. Challenges of creating and using code-mixed datasets are well-understood (Jamatia et al., 2016).

Towards this, we present an approach that uses deep learning for hate speech detection. We compare our approach with the past work by Bohra et al. (2018) and report a substantial improvement. The contribution of our work is:

1. We compare our deep learning-based approach with a statistical approach, and evaluate it on the same dataset as the statistical approach. We observe an improvement in the performance.

2. Instead of using pre-trained word embeddings, we train word embeddings on a large corpus of relevant code-mixed data. We demonstrate that this results in improved similarity values.

The rest of the paper is organised as follows. We describe related work in Section 2. The architecture is in Section 3 while the experiment setup is

[1] https://en.wikipedia.org/wiki/Hindi

[2] https://en.wikipedia.org/wiki/List_of_languages_by_number_of_native_speakers_in_India

[3] https://www.thehindu.com/news/national/centre-moves-for-law-on-online-abuse/article23295440.ece

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 155–160

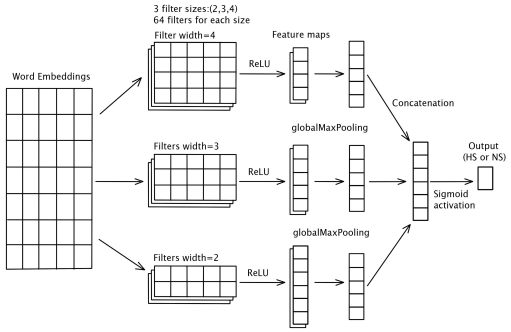


Figure 1: CNN model for hate speech detection.

in Section 4. We present our results in Section 5, and analyse the errors in Section 6.

2 Related Work

Approaches for hate speech detection have been reported (Schmidt and Wiegand, 2017; Warner and Hirschberg, 2012). Code-mixed datasets for Indian languages have been explored for several NLP tasks such as part-of-speech tagging (Jamatia et al., 2015), language identification (Das and Gambäck, 2014) and so on. Also, work concerning with hate speech in English language exists (Waseem and Hovy, 2016; Djuric et al., 2015; Davidson et al., 2017; Nobata et al., 2016). In a way, code-mixed datasets represent a majority of datasets from India, on the social media. Bohra et al. (2018) introduces a dataset of Hindi-English code-mixed tweets, and reports results on a statistical approach that use hand-engineered features. We download tweets from their dataset and compare with their results. Another work by Mathur et al. (2018) uses deep learning for hate speech detection. Our work differs from theirs in two ways: (a) We experiment with a different dataset, and compare performance on that dataset with the past work that reports results on the dataset, (b) We use domain-specific word embeddings that we show to be better indicative of semantics in the hate speech context. Our approach of using domain-specific embeddings is motivated by Tkachenko et al. (2018). They train two sets of word embeddings: one from a Wikipedia corpus and another from an Amazon review corpus. For sentiment-related tasks (such as sentiment classification), embeddings on the Amazon review corpus result in a higher performance as compared to those from the Wikipedia corpus. On the other hand, for topic-related tasks (such as topic classification), embeddings trained using the Wikipedia corpus outdo those from the Amazon review corpus.

3 Architecture

We propose three deep learning models for hate speech detection. These models are shown in Figures 1, 2 and 3 respectively. In the forthcoming sections, we describe each of the models.

3.1 CNN-1D

Figure 1 shows the CNN-1D model. It is fed in with domain-specific embeddings corresponding to sentences in the training data. The filters(3 filter sizes) with the specifications listed, convolve over the embeddings and produce the feature maps. Following this, we use a layer of globalMaxPooling having a dropout probability of 0.5. Then, the results are concatenated to form a single feature vector. Here, we apply the sigmoid activation to produce our final results.

3.2 LSTM

Figure 2 shows the LSTM model. Owing to the sequential nature of the code-mixed data, we make use of the LSTM model to compare our results. The results of the input embeddings, on passing through the LSTM layer, are made to accumulate at each proceeding timestep. The model is tuned to return the sequences of each of these timesteps. Next, the compiled sequences are given as an input to the globalMaxPooling layer. Lastly, the resulting output from the pooling layer is passed through the sigmoid activation function to give a final prediction.

3.3 BiLSTM

Figure 3 shows the BiLSTM model. Taking into consideration that the temporal dynamics can be better captured when a piece of text is analysed from both the directions, we make use of the BiLSTM to further compare our results. Here, instead of retrieving the sequences from a single direction, we do it for both the directions and concatenate the results. The vector now produced, goes through the globalMaxpooling layer. Finally, the result produced, is passed through the sigmoid activation to generate the final output.

3.4 Creation of Domain-Specific Word Embeddings

Using the Twitter API, we search for tweets containing Hindi cuss words and names of minority

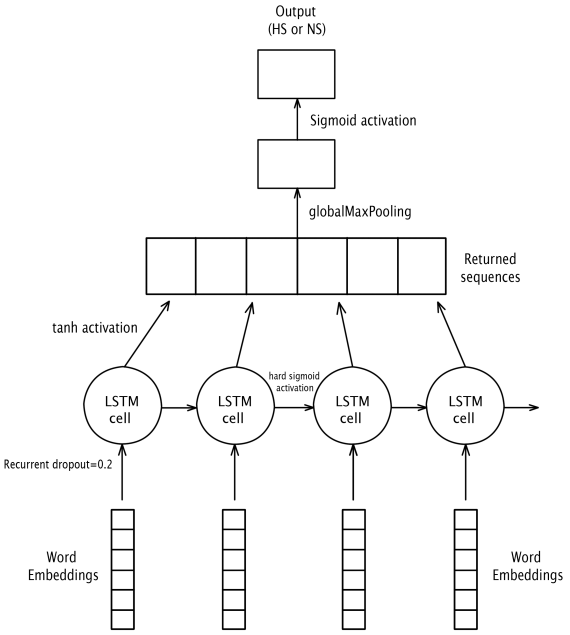


Figure 2: LSTM model for hate speech detection.

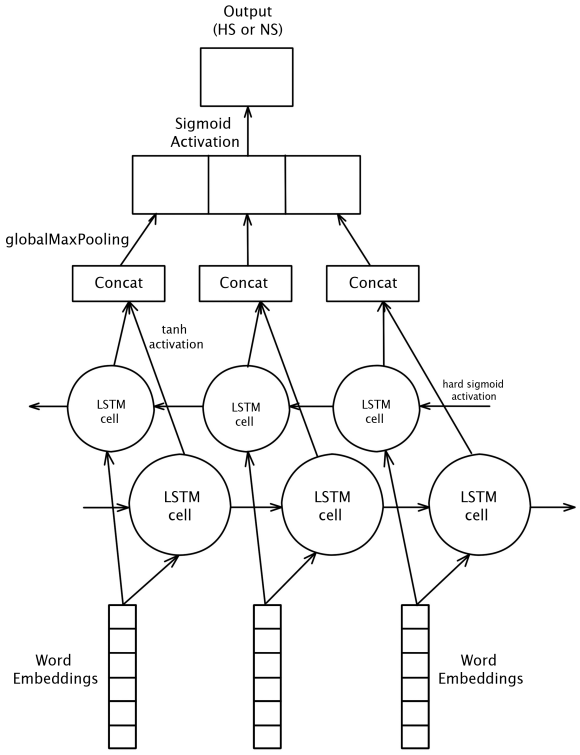


Figure 3: BiLSTM model for hate speech detection

Dataset Charecteristics	Size
Number of Tweets	255,309
Number of Timelines Extracted	7232
Number of Retweets	76,645
Total Number of Words	4,975,642
Size of Vocabulary	168,638
% Hindi Words per Tweet	18.63%

Table 1: Dataset Statistics of the Domain-Specific Word Embeddings

groups in their transliterated form. This is motivated by the definition of hate speech: hateful language that is used towards minority groups. We download a dataset of 255,309 tweets. Statistics of the dataset are in Table 1. Tweets collected were used only to train word embeddings. The dataset by Bohra et al. (2018) is used for evaluation of the 3 deep learning models.

We use the gensim (https://radimrehurek.com/gensim/models/word2vec.html) library to train word embeddings from this dataset, and use these domain-specific embeddings to initialize our deep learning models. We also utilize the Google Translate API to measure the average Hindi proportion of all the collected tweets. Using the API, we calculate the number of Hindi words in a tweet and calculate it's percentage with respect to the total number of words in the tweet. This is done for all tweets and an average is computed. *We commit to make our domain-specific word embeddings available for download at: https://github.com/satyaSK/Hate-Speech-Detection.*

4 Experiment Setup

We download the dataset by Bohra et al. (2018) using the Twitter API. Due to typical issues such as timeline restrictions, we obtain *3849* tweets, of which *1436* are labelled as hateful. We report 10-fold cross-validation performance on this dataset. We compare our models with a baseline re-implementation as given in Bohra et al. (2018). We implement feature extraction and use classification algorithms as described in their paper.

For the deep learning models, we use Keras, a neural network API (https://keras.io/). We experimentally determine the values of the parameters. For the CNN-1D model, we use the following hyperparameters:

1. Embedding dimension = 300
2. Number of filters of each filter size = 64, Batch size = 64, Epochs = 5, Dropout = 0.5
3. Pooling layer : Global max pooling
4. Filter sizes being 2,3 and 4 for the 3 CNNs in parallel.
5. Loss function : Binary cross-entropy loss
6. ReLU activation to obtain feature maps
7. Optimization algorithm : Adam

For LSTM and BiLSTM, we use the following configuration:

1. Number of LSTM units = 100, Recurrent dropout = 0.2
2. Loss function : Binary cross-entropy loss
3. Recurrent Activation : Hard sigmoid
4. Activation : tanh

We report Precision, Recall, F-score and accuracy values using methods in scikitlearn(Pedregosa et al., 2011).

5 Results

5.1 Qualitative Evaluation of domain-specific word embeddings

Table 2 shows cosine similarity between 'women' and words of three minority groups: religious, caste and sexual. We have not mentioned the specific names of the corresponding groups due to their controversial nature. *We wish to highlight that the word 'women' is used as a reference word solely because women might be a target of hate speech on social media*. Each row in the table is computed using the cosine similarity between the word 'women' and representative words of the specific minority group. The similarity between a pair of related social groups is consistently higher in the case of domain-specific embeddings as compared to general embeddings. For example, in case of sexual minority (which we consider as '*transgender*'), the similarity in the case of domain-specific embeddings is 0.726 while that in case of general embeddings is 0.348. This implies that domain-specific embeddings are able to capture the societal relationships and correlations between minority groups more accurately. An additional point to note is that, swear words in Hindi may not be present in pre-trained Google news embeddings. Specifically, we observe that 18 swear words in Hindi that were used to download the dataset, and were used to train domain-specific embeddings are not present in the Google news embeddings at all.

Minority Group	Domain-specific	General
Religious Minority	0.637	0.224
Caste Minority	0.615	0.204
Sexual Minority	0.726	0.348

Table 2: Cosine Similarity of 'women' with words representing three minority groups.

Therefore, higher similarity between groups that are targets of hate speech and higher coverage in terms of words that indicate expressions of hate, highlight the importance of using domain-specific embeddings.

5.2 Quantitative evaluation of hate speech detection

Bohra et al. (2018) train their classifiers using SVM and Random Forest algorithm, but only report accuracy. For a better comparison, we re-implement their features and obtain Precision, Recall and F-score values as well. The reported values and our values are compared with the deep learning models in Table 3. It must be noted that the accuracy values as reported and as obtained from re-implementation are close - indicating that the precision and recall are also likely to be comparable. We observe that using CNN-1D results in the highest performance with a F-score of 80.85% and an accuracy of 82.62%. This improvement in F-score is about 12% higher than the statistical baseline that we compare against. The improvement is in both precision and recall. An example of a correctly classified instance of hate speech by the CNN-1D model is '@.. *inke 6month ke works dekh lijiy nafrat ho jayegi aapko inse anandpal ke liye julus aur julus me public ko khule aam patthro ki barish karna dhamkana public ke sir fodna hate all of u*' which is translated to '@.. *look at the 6 month works of these people, you will start to hate them. A group of people rallying for Anandpal, has been stone-throwing and threatening the public. hate all of you.*'. Among the deep learning models, we observe that CNN-1D results in the highest precision while BiLSTM gives the best recall by a difference of approximately 0.40% as

	P (%)	R (%)	F (%)	A (%)
(Bohra et al., 2018) (SVM)	74.94	63.15	68.54	71.03 (71.7*)
(Bohra et al., 2018) (Random Forest)	62.43	58.88	60.60	65.78 (66.7*)
CNN-1D	**83.34**	78.51	**80.85**	**82.62**
LSTM	81.11	75.80	78.36	80.21
BiLSTM	82.04	**78.90**	80.43	81.48

Table 3: Comparison of Statistical Approach with Our Deep Learning-based Approach for Hate Speech Detection; * indicates reported values in the baseline paper; P: Precision, R: Recall, F: F-score, A: Accuracy.

compared to the CNN-1D. For example, this tweet '*@.. he is right x y may gundo ka palka kutha hai jo koi karwai nai kartha gundo par u.p no1 state in muders rape*' (*@.. he is right, x is a dog pet by the mafias of y, and so, he does not call for the investigation of the crimes they committed. u.p is number 1 state in rape and murders)* has been correctly classified as hate speech by the CNN-1D model while the LSTM and BiLSTM models incorrectly classify the tweet as non-hate speech.(x and y are anonymised names of a politician and a state respectively). In general, these results show that our deep learning models outperform the statistical approach.

6 Error Analysis

To understand the shortcomings of our models, we analyse and elucidate the errors made by our best-performing approach, which motivate future directions of experimentation. Some of these errors include:

- **Code-switched tweets in Hindi**: These are tweets written, following the grammatical structure of Hindi with a few English words. Many mis-classified examples include such tweets. An example is '@.. @.. @.. @.. *aur tum jahan hoti ho wahan balatkar badh jata hai baba bhi rape karne lagte hin (sic)*'. This tweet is translated as '@.. @.. @.. @.. *and rape cases start to increase wherever you go, baba also starts to rape'*. This has been identified to be a recurring error which occurs due to the code-mixed nature of the data at hand, where the text piece contains an imbalance between tokens from the Hindi and English scripts.
- **Series of swear words**: Some mis-classified instances are a string of swear words with a few function words between them. We skip an example here, on purpose, due to the obscene nature of these tweets. These errors may be because the model does not solely rely on the presence of swear words. Other context may be necessary to detect hate speech. This shows that the presence of explicit hate keywords or swear words is not the only determining factor for deciding whether a piece of text is hate speech or not, which points towards the necessity of capturing the underlying semantics and sense of the text in discussion.
- **Possibly incorrect labels**: Some tweets contain swear words but are not hateful towards any group as such. So, even though our models predict them as non-hate-speech, the instance is marked as mis-classified. For example, a hateful tweet calls someone the child of a rape victim but the gold label is negative. On the other hand, '*x ke samay me isase double rape hote the lekin us samay y bolti thi hai na (In times of x, the number of rapes were double as this, but y would always call it out, isn't it?!)* has the gold label as positive. (x and y are anonymised names of politicians).

7 Conclusion & Future Work

This paper explored hate speech detection in Hindi-English code-mixed tweets. We used three typical deep-learning models for detecting hate speech and empirically demonstrated their effectiveness. In contrast to statistical methods, our models were able to better capture the semantics of hate speech along with their context. We additionally demonstrated the efficacy of domain-specific word embeddings in adding intrinsic value to the code-mixed landscape.

Our work uses a benchmark dataset, and shows how deep learning models improve best-known work using statistical classifiers. In that, we make a small contribution to hate speech detection for Hindi-English code-mixed tweets. Novel deep learning techniques capable of assimilating textual cues more accurately, can be used to improve upon our work. Other nuances of hate speech in terms

of sarcasm or misinformation can also be incorporated in future work.

References

Kalika Bali, Jatin Sharma, Monojit Choudhury, and Yogarshi Vyas. 2014. " i am borrowing ya mixing?" an analysis of english-hindi code mixing in facebook. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 116–126.

Aditya Bohra, Deepanshu Vijay, Vinay Singh, Syed Sarfaraz Akhtar, and Manish Shrivastava. 2018. A dataset of hindi-english code-mixed social media text for hate speech detection. In *Proceedings of the Second Workshop on Computational Modeling of Peoples Opinions, Personality, and Emotions in Social Media*, pages 36–41.

Amitava Das and Björn Gambäck. 2014. Identifying languages at the word level in code-mixed indian social media text.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. *arXiv preprint arXiv:1703.04009*.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th international conference on world wide web*, pages 29–30. ACM.

Anupam Jamatia, Björn Gambäck, and Amitava Das. 2015. Part-of-speech tagging for code-mixed english-hindi twitter and facebook chat messages. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 239–248.

Anupam Jamatia, Björn Gambäck, and Amitava Das. 2016. Collecting and annotating indian social media code-mixed corpora. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 406–417. Springer.

Puneet Mathur, Rajiv Shah, Ramit Sawhney, and Debanjan Mahata. 2018. Detecting offensive tweets in hindi-english code-switched language. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 18–26.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153. International World Wide Web Conferences Steering Committee.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Maksim Tkachenko, Chong Cher Chia, and Hady Lauw. 2018. Searching for the x-factor: Exploring corpus subjectivity for word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1212–1221.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26. Association for Computational Linguistics.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Semi-Supervised Confidence Network aided Gated Attention based Recurrent Neural Network for Clickbait Detection

Amrith Rajagopal Setlur
amrithsetlur22@gmail.com

Abstract

Clickbaits are catchy headlines that are frequently used by social media outlets in order to allure its viewers into clicking them and thus leading them to dubious content. Such venal schemes thrive on exploiting the curiosity of naive social media users, directing traffic to web pages that won't be visited otherwise. In this paper, we propose a novel, semi-supervised classification based approach, that employs attentions sampled from a Gumbel-Softmax distribution to distill contexts that are fairly important in clickbait detection. An additional loss over the attention weights is used to encode prior knowledge. Furthermore, we propose a confidence network that enables learning over weak labels and improves robustness to noisy labels. We show that with merely 30% of strongly labeled samples we can achieve over 97% of the accuracy, of current state of the art methods in clickbait detection.

1 Introduction

With the number of social media users increasing by the day, one of the prime objectives of online news media agencies is to lead social media users onto bogus pages through the display of luscious text/images (Loewenstein, 1994). In most cases the content on the landing page is disparate to the headline the user clicked on. Source verification is no longer warranted as news agencies aren't held accountable for the content they post online. As (M. Potthast and Hagen, 2016) suggests, at least 15 of the most prominent content creators use clickbaits in some form or the other to honey-trap users. Impetus for such schemes can range from directing traffic to web sites that force users to purchase a product, to shaping popular opinion especially during elections. Some clickbaits claim to accomplish inconceivable tasks while others rely on a viewer's inducement to grapevines.

- "You will never believe what this celebrity did at the awards ceremony."
- "10 things that will get you fairer in 5 days."
- "Millionaires want to conceal this scheme. It can make you rich in a week."

Earlier approaches on tackling clickbaits mainly focused on cheap and easy to implement solutions. Blacklisting URLs has been, to some extent, effective in regulating an average user's exposure to clickbaits. (Gianotto, 2014) assumed that most clickbaits are based on a few key phrases, and a naive way yet effective strategy would be to simply look for such phrases. Such an assumption holds no ground today. As the problem grew to be more pervasive, social media companies modeled the probability for a content to be a clickbait based on factors like click-to-share ratio, time spent by user on the target page, and other such quantifiers. Recent research focuses on salvaging sentence structures, n-grams & embeddings among other features, in classifiers like Random Forests (RF), Gradient Boosted Trees (GBT), Support Vector Machines (SVM) or the vanilla Logistic Regression (LR). With the advent of online news agencies, there exists a plethora of such sources, but labeling each of the headlines from them would be a staggering task. This vindicates the need for an unsupervised / semi-supervised approach.

Contributions of this paper include: 1. A novel loss component on the attention weights that encodes prior information from a weak source of la-

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 161–170

bels, which eventually improves the generalizability of the deep learning model that has been trained on a small representative dataset. 2. A joint architecture that incorporates into the clickbait classification framework a confidence network that tackles label noise. 3. Using Gumbel-Softmax for gated attentions, thus enforcing peaky attentions over word contexts. 4. Empirically proving the performance of the proposed approach on popular clickbait datasets with only a small portion of labeled samples.

2 Related Work

The problem of clickbait detection has been primarily studied in two forms. One of them involves a pair of post and target phrases, in which, the objective is to identify whether the post text (visible to the user), is in someway related to the target content (text/images on the landing page). Using this formulation, (P. Biyani and Blackmer, 2016) suggested that the features involved in clickbait detection can be broadly classified into: content features (quotes, capitalization), similarity between the source and target representations, informality, forward reference, URLs etc. A gradient boosted tree was trained using these features. (Chakraborty et al., 2016) highlighted the use of features based on linguistic and structural differences between the clickbait & non-clickbait headlines. Using the n-grams and word patterns, they successfully trained a SVM classifier with a Radial Basis Function (RBF) kernel, that outperformed the baseline rule based method in (Gianotto, 2014).

The other form comprises of making decisions purely based on the headline content (Zhou, 2017), (P. Biyani and Blackmer, 2016). Our work is based on this paradigm, and performs on par with methods that follow the former approach. According to (Schuster and Paliwal, 1997), Recurrent Neural Network (RNN) based sentence embeddings of the headlines, are expressive enough to learn neural network based classifiers that separate the two classes. (Teng et al., 2017) explored convolutional networks that can convolve our word embeddings to learn n-grams, sub-words and token patterns that are consistent with clickbaits. (M. Potthast and Stein, 2017) worked on the twitter dataset and treated clickbait detection as a regression problem instead of binary classification. Hence, they proposed a model that outputs scores indicative of how clickbaity a tweet is. (A. Anand and Park, 2017) used a bi-directional LSTM to improve upon the results published by (Chakraborty et al., 2016), on the *Headlines Dataset* (Section 3.1).

(Wang et al., 2016) employed attention based mechanisms (Bahdanau et al., 2014) for text classification. Our work reuses the self-attentive structured sentence embeddings introduced by (Z. Lin and Bengio, 2017), with some additional components that supplant text classification in semi-supervised environments, with weakly labeled data.

Some of the most recent approaches that have been successful in modelling curiosity are based on the "novelty" and "surprise" components of a headline. (Venneti and Alam, 2018) used topic modelling to identify the topic distributions for each headline in the corpus. Distance metrics in the space of propability distributions like KL divergence and Hellinger distance were used as features to express novelty while surprise was primarily modeled using bi-gram frequency counts. Based on these features, an SVM/LR classifier was trained on a small section of the training data.

3 Proposed Methodology

3.1 Dataset

The first dataset used in the paper is the *Headlines Dataset* curated by (Chakraborty et al., 2016) [1] and used by (A. Anand and Park, 2017), (Rony et al., 2017) et al. It holds labels for 32,000 headlines which featured in articles from BuzzFeed, New York Times, Scoopwhoop, Upworthy, The Guardian, ViralNova, The Hindu, ViralStories, Thatscoop and WikiNews. In all, there are 15,999 clickbait and 16,001 non-clickbait samples. We perform an ablation study using a varying proportion of the dataset as our strongly labeled set. Furthermore, we compare our results against strong baselines established on this dataset, in the absence of labeled data. All experiments have been performed using a 5-fold cross validation scheme, in order to reconcile with the existing baselines.

The second dataset was picked from the *Clickbait-Challenge 2017* (Zhou, 2017)[2]. The challenge posed clickbait detection as a regression problem, assigning each entry a set of scores from five different annotators. Each $score \in$

[1] https://github.com/bhargaviparanjape/clickbait/

[2] https://www.clickbait-challenge.org/

$\{0, \frac{1}{3}, \frac{2}{3}, 1\}$[3]. Therefore, each sample was annotated with score summary statistics: truthMean, truthJudgements, truthMode, and truthMedian. For the sake of consistency we reformulated this as a classification problem using the following decision:

$$C_i = \begin{cases} clickbait & \text{if } truthMean_i \geq 0.5 \\ non-clickbait & otherwise \end{cases} \quad (1)$$

Although the dataset consists of "targetText" and "image" (landing page) data apart from "postText" (headline), we were able to attain convincing results by using features derived purely from "postText". This assumption was based on the behaviour of an average annotator, as in most cases the annotator judgments are purely hinged on the headline. Based on similar assumptions, (A. Anand and Park, 2017), (Zhou, 2017) and (Rony et al., 2017), used n-grams, simple word filters or latent text representations (LSTMs) that were solely based on headline content. Following 3 splits were provided in the Clickbait-Challenge (C: Clickbaits, NC: Non-clickbaits).

Label	Headlines	C	NC
A	19,538	4,761	14,777
B	2,495	762	1,697
C	80,012	-	-

We evaluated F1-score and accuracy on the test set B while using portions of the set A as our labeled dataset (with the remaining as unlabeled), bootstrapped with the unlabeled set C.

3.2 Random Forest

In order to identify words of high importance (information gain), we trained a Random Forest Classifier on the strongly labeled section of the dataset. By using entropy (Eq. 2) as a measure of information gain, while splitting samples at each node of a tree, we posited that words with low entropy (summarized in Table 1) were strong signals for identifying clickbaits. Before fitting the random forest, the headlines were pre-processed; all numeric content was mapped to <n-token>, URLs were mapped to <u-token>. Along with these, entity detectors were useful in identifying references to dates/years, locations. The Wordnet Lemmatizer was used to obviate trivial variances in word representations. The analyses presented in Table 1 was done on the *Headlines Dataset*. In this section, we propose the use of Random Forest as a source of weak labels, with the Random Forest being trained on the human labeled samples.

[3]https://www.clickbait-challenge.org/#data

Word	Importance	Naive Inclination
<n-token>	5.120	clickbait
like	3.112	clickbait
dies	3.042	non-clickbait
people	2.351	clickbait
know	2.336	clickbait
life	2.155	clickbait
need	2.062	clickbait
president	1.799	non-clickbait
wins	1.664	non-clickbait
kill	1.623	non-clickbait
iraq	1.244	non-clickbait
hilarious	1.058	clickbait
favorite	1.039	clickbait
laugh	0.965	clickbait
really	0.963	clickbait
court	0.941	non-clickbait
china	0.940	non-clickbait
dead	0.891	non-clickbait
photos	0.869	clickbait
most	0.851	clickbait
leader	0.702	clickbait
pictures	0.698	clickbait
obama	0.592	non-clickbait
questions	0.524	clickbait

Table 1: Some of the tokens with high word-importance factor (Inverse Entropy).

$$E_s = \frac{N_l}{N} E_l + \frac{N_r}{N} E_r$$
$$E_l = -\sum_{i \in C} p_{i_l} \log p_{i_l}$$
$$E_r = -\sum_{i \in C} p_{i_r} \log p_{i_r} \quad (2)$$

p_{i_l}: proportion of samples of the left split that $\in$ class C_i
p_{i_r}: proportion of samples of the right split that $\in$ class C_i
N_l: number of samples in left split
N_r: number of samples in right split
N: total number of samples

Salvaging the tokens with the lowest entropy, we built simple rules to detect clickbaits. Tree paths that included decisions based on these low entropy tokens were used to construct rules in Disjunctive Normal Forms (DNFs) (Table 2). The unlabeled dataset was then run through these rules to determine weak labels for classification. This aided the training of the attention network, as corroborated by our experiments (Section 4).

3.3 Problem Definition

We are given two sets of data, namely: $D_s = \{(X_1, y_1), (X_2, y_2), .., (X_n, y_n)\}$, that is strongly labeled through manual annotations and $D_w = \{(X_1, \hat{y}_1), (X_2, \hat{y}_2), .., (X_n, \hat{y}_n)\}$, which is weakly labeled and is composed of samples from the unlabeled set. The weak labels in D_w are generated by the Random Forest Classifier (RF) (Section 3.2). In addition to this, we assimilate $\hat{D}_s$, that is composed of RF predictions on D_s (Eq. 4).

$$\begin{gathered} \hat{y}_i = RF(X_i) \\ \forall (X_i, \hat{y}_i) \in D_w \end{gathered} \tag{3}$$

$$\begin{gathered} \hat{D}_s = \{(X_i, y_i, RF(X_i))\} \\ \forall (X_i, y_i) \in D_s \end{gathered} \tag{4}$$

The goal would be to train a classification network on the set obtained by concatenating D_s & D_w. It is assumed that D_s is comprised of a representative set of samples and that D_s and D_w contain i.i.d. samples from the true data distribution. Since D_w consists of weak labels from the Random Forest and $|D_w| > |D_s|$, we propose the use of a confidence network (Section 3.7) that predicts the accuracy of a weak label. These accuracy scores would re-weigh the gradient updates when the loss is calculated using samples from D_w, thus attenuating the effect of noisy labels.

3.4 Deep Attention Network

Self-attentive structured attention mechanisms for efficient semantic latent sentence representations was proposed by (Z. Yang and Hovy, 2016) and (Z. Lin and Bengio, 2017). (Zhou, 2017) further ascertained the effectiveness of the attention mechanism in clickbait detection. The intuition behind a self attention mechanism is that the network learns the importance of each token's context, for the task in hand (clickbait detection), along with a hidden state representation of the word context itself. While (Zhou, 2017), (Z. Yang and Hovy, 2016) used this in a fully supervised setting, we researched its resilience under the semi-supervised tone. In the following sections, we show that, the presence of an external attention enforcer is pivotal in training the network. Merely training a deep network on a meagre set of a few thousand samples leads to overfitting (Goodfellow et al., 2016), and as expected, this claim was substantiated, when we noted a higher validation/test loss, upon direct application of the architecture proposed by (Zhou, 2017), (Z. Lin and Bengio, 2017).

We propose vital and consequential modifications to the base network which accommodates the semi-supervised learning problem. Attention mechanisms have traditionally been utilized to focus attention on features that are most influential in performing a specific task, like image captioning or semantic segmentation, in the image domain (Xu et al., 2015). Drawing a parallel, we propose an attention based loss that forces the attentions for a specific set of words to be higher than the rest. When a large number of samples are available, the attention module is self-sufficient in determining such tokens. In this case however, we use as a surrogate, the word importance measures from the Random Forest classifier to identify them (Section 3.6).

3.5 Architecture

Figure 1 demonstrates the two networks employed during training. The classification/self-attention network determines the sentence embedding post a dot-product operation on the LSTM hidden state representations using the attention weights. Given a headline with N tokens, we map each token w_i, where $i \in \{1, \ldots, N\}$, to its corresponding glove embedding[4] (trained on the twitter corpus), denoted by x_i. A bi-directional LSTM network, encodes the word context (from both directions), for the word w_i, in the time step t_i (Eqs. 5).

$$\begin{gathered} h_{left_i} = LSTM_{left}(x_i) \\ h_{right_i} = LSTM_{right}(x_i) \\ h_i = h_{left_i} \,||\, h_{right_i} \\ x_i \in R^e \\ h_{left_i}, h_{left_i} \in R^u \\ h_i \in R^{2u} \end{gathered} \tag{5}$$

Given the word context embeddings, the at-

[4] https://nlp.stanford.edu/projects/glove/

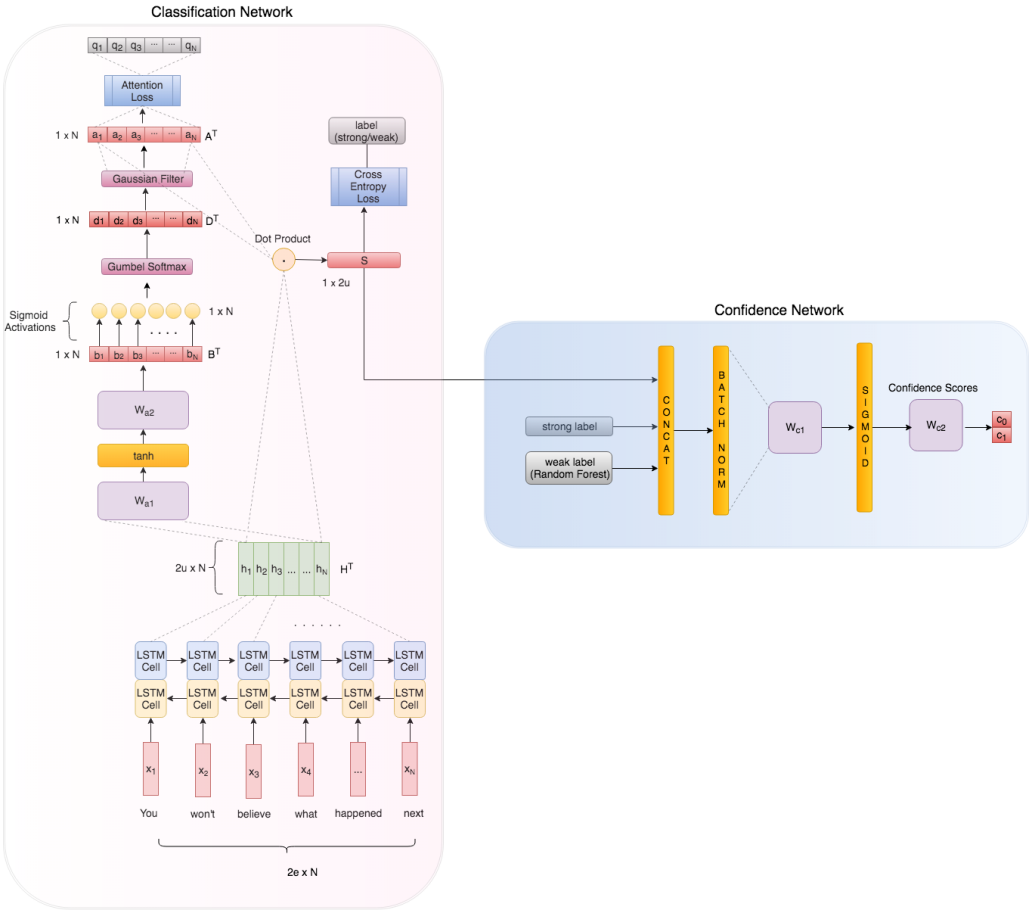


Figure 1: Network Architecture

tention network maps them to intermediate token level activations, using a Multi-Layered Perceptron (MLP) network with non-linear (tanh) activations. With $H \in R^{N \times 2u}$ as the context embedding matrix, the network outputs $B \in R^N$, as the intermediate activation vector (Eq. 6).

$$B = \sigma((\tanh(H \cdot {W_{a1}}^T)) \cdot W_{a2})$$
$$H \in R^{N \times 2u},\ W_{a1} \in R^{m \times 2u},\ W_{a2} \in R^{m \times 1} \quad (6)$$

The original structured self-attentive network (Z. Lin and Bengio, 2017) proposed a softmax activation in the final layer. This was befitting in the case of sentiment classification, where in the sentiment of each sentence was pivoted on a few key tokens and such tokens existed in all sentences, irrespective of the class. On the contrary, in the case of clickbait detection, it is mostly the negative class (clickbaits), that contains such tokens of importance. As depicted in Table 1, the words identified by the Random Forest for the clickbait class are independent of the news item/news subject ("<n-token>", "know", "need", "favorite", "most"). Same can't be said for the positive class ("obama", "iraq", "china", "president"). Hence we introduced a sigmoid layer to better suit the case of clickbait detection.

The intermediate activations $B \in R^N$, obtained post the sigmoid layer were treated as parameters of a binary distribution. Treating each b_i as $P(a_i = 1)$, we could sample from the binary distribution. To propagate losses during back-propagation, we instead sampled values from the Gumbel-Softmax distribution (Jang et al., 2016), to obtain $D \in R^N$ (Eq. 7). This encapsulates the "gated" portion of the network where in activations for word contexts are sampled from a discrete distribution whose parameters are learned. The central idea is to support a case where in the sentence may not have significant impact words (fairly common with the positive class), and with this formulation the sampled values can represent such a case trivially with a zero vector. In the original work done by (Z. Lin and Bengio, 2017), such activations were inconceivable.

$$d_i = \frac{\exp\left((\log(b_i) + g_{i_1})/\tau\right)}{\exp\left((\log(b_i) + g_{i_1})/\tau\right) + \exp\left((\log(1 - b_i) + g_{i_0})/\tau\right)} \tag{7}$$

$g_{i_1} \sim$ Gumbel(0, 1)
$g_{i_0} \sim$ Gumbel(0, 1)
τ: the temperature parameter that determines the extent to which d_i would be close to 0 or 1

Adding the Gumbel-Softmax enables the possibility of peaky activations. Lower values of τ would lead to d_i lying on either end of the spectrum, i.e 0 or 1, thus resembling a sample from a binomial distribution.

In cases of very low volumes of labeled data we observed that the attention weights of tokens neighboring the words of importance were very close to 0, and thus their context representations wasn't contributing towards the final sentence representation. To avoid this, we used a Gaussian filter over the samples from the Gumbel-Softmax distribution, which acts as a 1D convolution. The weights of the Gaussian filter are not learned. They are fixed and normalized so as to ensure that the final attention weights $0.0 \leq a_i \leq 1.0$. The activation vector $A \in R^N$, and LSTM hidden states H produce the final sentence embedding S (Eq. 8).

$$\begin{gathered} A = GaussianFilter(B) \\ S = H^\top \cdot A \\ H \in R^{N \times 2u},\ A \in R^N,\ S \in R^{2u} \end{gathered} \tag{8}$$

3.6 Attention Loss

One way to identify key words/features is to use a weak labeler like Random Forest. It identifies a set of words crucial in decision making (Table 1). The loss penalizing a deviation from such important words, is the standard binary cross entropy loss where it considers the attentions sampled from the Gumbel-Softmax distribution. The true attention is inferred from the word importance or entropy scores generated by the Random Forest. A true attention of 1.0 is assigned to words with importance over a particular threshold.

The set of positive activations in the corpus is much lower than its counterpart. This class imbalance was tackled by simply using the class weights to re-weigh the attention loss (Alejo et al., 2017). The proportion of activations of class i is inversely proportional to w_i (Eq. 9). The parameters of the classification network in Figure 1 are optimized over a combination of the attention and classification losses (Eq. 10).

$$\begin{gathered} \mathcal{L}_a(X, y) = -W^\top [Q \odot log(A) + \bar{Q} \odot log(\bar{A})] \\ A = \{a_1, \ldots, a_N\}^\top,\ \bar{A} = 1 - A \\ Q = \{q_1, \ldots, q_N\}^\top,\ \bar{Q} = 1 - Q \\ W = \{w_1, \ldots, w_N\}^\top \end{gathered} \tag{9}$$

A: network activations for tokens
Q: true activations for tokens
W: sample weights
N: number of tokens
$\odot$: Hadamard product
X, y: samples $\sim D_s, D_w$

$$\mathcal{L}_{ca}(X, y) = \mathcal{L}_c(X, y) + \lambda \cdot \mathcal{L}_a(X, y) \tag{10}$$

X, y: samples $\sim D_s, D_w$
$\mathcal{L}_{ca}$: Classification + Attention Loss
$\mathcal{L}_c$: Classification Loss
$\mathcal{L}_a$: Attention Loss
λ: Contribution of the attention loss to the total loss

3.7 Optimization Algorithm

Amalgamation of strong and weak supervision has been used to solve constraints like data paucity, noisy labels and a few others (Schapire, 1990). Weak learners are sources of noisy labels. (Dehghani et al., 2017b) proposed the use of a confidence network to estimate the accuracy of a noisy label. Similar to (Dehghani et al., 2017a) and (Arachie and Huang, 2018), we used an optimization method that is a variant of the standard Stochastic Gradient Descent (SGD). The latter uses a constant learning rate on all samples in an iteration. Such an approach can lead to noisy gradients, especially when the proportion of strongly labeled samples in a batch is low (Goodfellow et al., 2016).

In order to mitigate this, a confidence network, trained on $\hat{D}_s$, formulates the confidence in a weak label as a function of the weak label (Random Classifier output) and the encoded input representation. As shown in Figure 1, the architecture is split into the classification and confidence networks. Batches of i.i.d samples are drawn iteratively from $\hat{D}_s$ and D_w. A batch of the former type is used train the classification network, the weights for which are updated using SGD. Since

the samples from $\hat{D}_s$ also consist of RF predictions on strongly labeled samples, they are used to train the confidence network (f_{conf}) as well.

$$\begin{aligned} score &= f_{conf}(E(X), \hat{y}) \\ &\forall (X, y, \hat{y}) \in \hat{D}_s, \\ E(X) &: latent\, representation\, for\, X \end{aligned} \tag{11}$$

For a sample $(X, y, \hat{y}) \sim \hat{D}_s$, a forward propagation on the classification network would generate the sentence representation S (denoted as $E(X)$ in Eq. 11). S is concatenated with $\hat{y}$, and passed to a batch-normalization layer. Since the embeddings and binary signals lie on separate manifolds, the batch-normalization is quintessential before the concatenated input is passed to a neural network. Figure 1 enunciates the output of the network as a 2-dimensional vector $[c_0, c_1]$. The $score$ in Eq. 11 is given by c_1. The confidence network is trained using a cross-entropy loss ($\mathcal{L}_{conf}$ in Eq. 12). The true confidence is given by the indicator $\mathbb{1}_{y=\hat{y}}$.

In the subsequent iteration when samples are drawn from D_w confidence scores are inferred on each sample in this set through a forward propagation on the confidence network. These scores are passed to the optimization algorithm for the classification network in order for it to re-weigh its gradient updates. Equation 13 defines the gradient update rule for parameters in the classification network.

$$\mathcal{L}_{conf}(X, y, \hat{y}) = -\mathbb{1}_{y \neq \hat{y}} \log c_0 - \mathbb{1}_{y=\hat{y}} \log c_1 \tag{12}$$

$(X, y, \hat{y}) \sim \hat{D}_s$
$c : [c_0, c_1]$, output of the confidence network
$\mathcal{L}_{conf}$: Loss on sample $(X, y, \hat{y})$
$\mathbb{1}_{y=\hat{y}}$: True confidence value

$$\widehat{\nabla w} = \frac{1}{B} \sum_{i=1}^{B} f_{conf}(X_i, \hat{y}_i) \cdot \nabla_w \mathcal{L}_{ca}(X_i, \hat{y}_i)$$
$$w_{t+1} = w_t - \eta_t \widehat{\nabla w} \tag{13}$$

η_t: learning rate
B: Batch size of samples drawn from $\hat{D}_s$
$\mathcal{L}_{ca}$: Loss on sample $(X_i, \hat{y}_i)$ (Section 3.6)
f_{conf}: Confidence network

Rule	**Class**
believe ∧ <n-token>	C
president ∧ iraq ∧ war	NC
hilarious ∧ photos ∧ <n-token>	C

Table 2: Rules drawn from the tress in RF.

4 Experiments

4.1 Experimental Setting

The following discussion on hyper-parameters is with respect to the *Headlines Dataset*. Although nearly the same set of values were applicable to the *Clickbait-Challenge Dataset*.

The hyper-parameters of the random forest were fine tuned using Bayesian optimization techniques. We used 50 estimators, with a maximum depth of 3 and a minimum split size of 5 along with the entropy criterion for splitting. Rules in DNF form were constructed by traversing paths that consisted only of tokens with high information gain. The threshold for minimum word importance was found to be optimal at 0.42. Table 2 lists some of the rules (mentioning only the words whose presence triggers the corresponding rule).

The dimensionality of the parameters entrenched in the classification and confidence networks have been encapsulated in Table 3. Glove embeddings (Pennington et al., 2014) trained on the twitter corpus, were used as base word embeddings, which were fed to the LSTM cells. Dropouts (Srivastava et al., 2014) have been commonly used in recurrent neural networks as a form of regularizer, especially after they were proven to be a form of Bayesian inference in deep Gaussian Processes (Gal and Ghahramani, 2016). We used a dropout parameter of 0.5 within the LSTM, and 0.4 for the inputs to the fully connected layer with weights W_{a2}. For the Gumbel-Softmax layer, temperature parameter of $\tau = 0.7$ was found to be optimal. Lower values of τ would have led to higher attention weights for the words of importance, but it would have also driven the weights for the rest of the tokens to zero. For the batch-normalization layer present within the confidence network, a momentum of 0.05 resulted in a slightly better validation accuracy as compared to the standard value of 0.1 (Ioffe and Szegedy, 2015). This can be attributed to the variance in label confidence across alternate mini-batches that were sampled randomly from the labeled and unlabeled sets. The

Parameter	Dimensionality
x (Word Embedding)	300
h (LSTM hidden state)	200
W_{a1}	32×400
W_{a2}	32×1
W_{c1}	64×400
W_{c2}[5]	65×2

Table 3: Dimensionality of the parameters in the classification and confidence networks.

standard mini-batch SGD optimizer with a learning rate of 0.0001 (Li et al., 2014) and a batch size of 64 samples was employed. The parameter λ, involved in the combination of attention and classification losses was fixed at 0.3. We used an early stopping criteria, to stunt training. In most cases, 5 epochs were sufficient to fit the training data, a result, which seems to corroborate with the size of the dataset involved.

4.2 Results

In accordance with the disparity of the datasets mentioned in section 3.1, we summarize the proposed model performance on the two of them independently. In both cases, we benchmark the model performance against baselines, that claim to have achieved state of the art results on the dataset in question.

(A. Anand and Park, 2017) claimed a high classification accuracy of 0.982 after having experimented with multiple RNN based architectures to embed the clickbait embeddings in a multi-dimensional space. Our model's performance emulates the former on the fully labeled dataset. On the partially labeled set we achieve a high accuracy of 0.980, even with only 30% of labeled samples (Table 4). This is an increment of **4.03%** in the accuracy on the validation set, when compared to the BiLSTM based network.

We further study model performances across the various model architectures supplanted in an incremental fashion (Table 5). When data paucity is high (30% labeled), we see significant differences in accuracy, precision and recall while adding individual components to the network. Increments have been noticed when a large number of labeled samples (80%) are available, albeit the rate of improvement is negligible. The gaussian filter over attention weights sampled from Gumbel-Softmax is more effective in the former case where scattering attention onto the neighborhood of high importance words prevents the sentence representation from collapsing to an average of word vectors in the inchoate stages of training. The precision and recall metrics also show similar trends, with increments of **3.92%** and **3.91%** respectively, in case of 30% labeled samples.

In case of the *Clickbait-Challenge* dataset (section 3.1), we train our model on the labeled split (A), concatenated with the unlabeled samples in (C) and observe model performance on the test set (B), consistent with (Zhou, 2017). We further performed experiments to study our model's ability to learn meaningful sentence representations when only a portion of the set A was labeled. Table 6 puts into perspective the observed Mean Squared Error (MSE), precision and accuracy on the test set, in comparison to the current best performing model (Zhou, 2017) on the dataset in question. The existing baseline involves sentence classification solely using the self-attention network introduced by (Z. Lin and Bengio, 2017). The baseline results are convincing when all labels are available and on par with our model's performance. On the other hand, when only 30% of the set A is used as the labeled set we observed a jump of **23.55%** & **25.61%** with regards to the accuracy & F1-score respectively.

Conclusion and Future Work

In this paper, we proposed a novel architecture to tackle clickbait detection when only a few labeled samples are present. We successfully showed that use of a weak labeler like Random Forest can generate priors over an attention mechanism and thus improve generalizability. Empirically, we have also shown that training a confidence network to rescale gradients, helps tackle the inherent noise attributed to the presence of a weak labeler.

We haven't considered λ as a function of time, and annealing λ over time may improve performance. Future work may also involve confidence networks with, momentum and adaptive learning rate based gradient update methods like Adam or RMSprop (Kingma and Ba, 2014). The glove embeddings capture the word contexts. Modelling curiosity (Venneti and Alam, 2018) in conjunction with such features may help capture user intent as well. Nevertheless, this requires a different set of experiments and benchmarks to thoroughly under-

[5]This includes the bias terms as well.

Model	Accuracy	Precision	Recall	ROC-AUC
Baseline (BiLSTMs)	0.982	**0.984**	0.980	**0.998**
Self-Attentive Network (SAN)	0.982	0.983	0.981	0.997
SAN + Gumbel Softmax (GS)	0.981	0.982	0.981	0.996
SAN + GS + Gaussian Filter (GF)	**0.983**	**0.984**	**0.982**	0.997

Table 4: Performance metrics against the existing baseline (A. Anand and Park, 2017), with the fully labeled *Headlines Dataset*. [Averaged over a 5-fold cross validation scheme]

Model	Accuracy			Precision			Recall		
	80%	50%	30%	80%	50%	30%	80%	50%	30%
Baseline (BiLSTMs)	0.980	0.966	**0.942**	0.979	0.967	**0.943**	0.981	0.966	**0.944**
SAN	0.979	0.966	0.944	0.978	0.965	0.945	0.980	0.967	0.942
SAN + RF	0.980	0.976	0.959	0.980	0.974	0.958	0.981	0.976	0.960
SAN + RF + GS	0.980	0.978	0.970	0.981	0.978	0.971	0.982	0.979	0.972
SAN + RF + GS + GF	0.981	0.977	0.975	0.982	0.978	0.976	0.981	0.977	0.977
SAN + RF + GS + GF + Confidence N/w	0.983	0.982	**0.980**	0.984	0.983	**0.980**	0.982	0.982	**0.981**

Table 5: Ablation study with variations in the proportions (80%, 50%, & 30%) of labeled data (*Headlines Dataset*). [Averaged over a 5-fold cross validation scheme]

Model	MSE			F1-Score			Accuracy		
	100%	50%	30%	100%	50%	30%	100%	50%	30%
Baseline	**0.033**	0.047	0.055	**0.683**	0.557	0.521	**0.856**	0.713	0.671
Proposed Model	0.034	**0.038**	**0.042**	0.679	**0.668**	**0.662**	**0.856**	**0.835**	**0.829**

Table 6: MSE, F1-Score and Accuracy metrics for the existing baseline (Zhou, 2017) & our solution on the test set, while using different proportions of set A (100%, 50% & 30%) as our labeled data.

stand the intricacies involved in such a mixture.

References

T. Chakraborty A. Anand and N. Park. 2017. We used neural networks to detect clickbaits: You wont believe what happened next! In *European Conference on Information Retrieval*, pages 541–547. Springer.

Roberto Alejo, Juan Monroy-de-Jesús, J. C. Ambriz-Polo, and J. H. Pacheco-Sanchez. 2017. An improved dynamic sampling back-propagation algorithm based on mean square error to face the multi-class imbalance problem. *Neural Computing and Applications*, 28(10):2843–2857.

Chidubem Arachie and Bert Huang. 2018. Adversarial labeling for learning without labels. *CoRR*, abs/1805.08877.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Abhijnan Chakraborty, Bhargavi Paranjape, Sourya Kakarla, and Niloy Ganguly. 2016. Stop clickbait: Detecting and preventing clickbaits in online news media. In *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*, pages 9–16. IEEE.

Mostafa Dehghani, Arash Mehrjou, Stephan Gouws, Jaap Kamps, and Bernhard Schölkopf. 2017a. Fidelity-weighted learning. *CoRR*, abs/1711.02799.

Mostafa Dehghani, Aliaksei Severyn, Sascha Rothe, and Jaap Kamps. 2017b. Learning to learn from weak supervision by full supervision. *CoRR*, abs/1711.11383.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1050–1059.

A. Gianotto. 2014. Downworthy: A browser plugin to turn hyperbolic viral headlines into what they really mean. *downworthy.snipe.net*.

Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. 2016. *Deep Learning*. Adaptive computation and machine learning. MIT Press.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *CoRR*, abs/1611.01144.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J. Smola. 2014. Efficient mini-batch training for stochastic optimization. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 661–670.

G. Loewenstein. 1994. The psychology of curiosity: A review and reinterpretation. *Psychological bulletin, 116(1):75*.

B. Stein M. Potthast, S. Kpsel and M. Hagen. 2016. Clickbait detection. In *European Conference on Information Retrieval*, pages 810–817. Springer.

M. Hagen M. Potthast, T. Gollub and B. Stein. 2017. The clickbait challenge 2017: Towards a regression model for clickbait strength. *Clickbait Challenge*.

K. Tsioutsiouliklis P. Biyani and J. Blackmer. 2016. 8 amazing secrets for getting more clicks: Detecting clickbaits in news streams using article informality. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 94–100.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Md Main Uddin Rony, Naeemul Hassan, and Mohammad Yousuf. 2017. Diving deep into clickbaits: Who use them to what extents in which topics with what effects? In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, Sydney, Australia, July 31 - August 03, 2017*, pages 232–239.

Robert E. Schapire. 1990. The strength of weak learnability. *Machine Learning*, 5:197–227.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45(11):2673–2681.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Ervin Teng, João Diogo Falcão, and Bob Iannucci. 2017. Clickbait: Click-based accelerated incremental training of convolutional neural networks. *CoRR*, abs/1709.05021.

Lasya Venneti and Aniket Alam. 2018. How curiosity can be modeled for a clickbait detector. *CoRR*, abs/1806.04212.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 606–615.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2048–2057.

C. N. d. Santos M. Yu B. Xiang B. Zhou Z. Lin, M. Feng and Y. Bengio. 2017. A structured self-attentive sentence embedding. In *5th International Conference on Learning Representations*.

C. Dyer X. He A. J. Smola Z. Yang, D. Yang and E. H. Hovy. 2016. Hierarchical attention networks for document classification. In *Conference of the - Human Language Technologies*, pages 1480–1489. ACL.

Yiwei Zhou. 2017. Clickbait detection in tweets using self-attentive network. *In Proceedings of the Clickbait Challenge*.

User Perception of Code-Switching Dialog Systems

Anshul Bawa **Monojit Choudhury** **Kalika Bali**
Microsoft Research
Bangalore, India
{monojitc,kalikab}@microsoft.com

Abstract

Bilingual speakers often freely mix languages in conversation. Should dialog systems also be designed with an ability to code-switch, when interacting with multilingual users? In this paper, we explore this question based on a user-study on text-based bot-human conversations. Our results reveal three distinct classes of users with varying individual attitude towards code-switching (CS), and demonstrate the importance of a bot's CS fluency and its ability to reciprocate CS, in determining user preference. We also highlight some computational and sociolinguistic considerations that have implications for the design of multilingual dialog systems, and propose a strategy for dialog systems to navigate attitude estimation in mixed-language interactions.

1 Introduction

Code-switching (CS) is the fluid alteration between two or more languages within a conversation, and is common in most multilingual societies (Gumperz, 1982; Myers-Scotton, 1993). Multilingual speakers are known to code-switch in casual speech conversations for reasons motivated by its socio-pragmatic functions (Auer, 2013a; Begum et al., 2016; Auer, 1995), and driven by communicative and cognitive principles (Myslín and Levy, 2015; Scotton and Ury, 1977). As a marker of a shared multilingual identity (Auer, 2005), CS can make a conversation sound more natural and engaging, convey informality, and reduce perceived social distance between speakers (De Fina, 2007; Camilleri, 1996; Myers-Scotton, 1995).

Text-based conversational agents are now being developed in new languages (Shum et al., 2018). Although a large fraction of the world's population is multilingual (Ansaldo et al., 2008), nearly all conversational agents are still monolingual, which begs the following questions: Should dialog systems be designed to understand and respond in code-switched languages as well, or is it sufficient to simply have multiple monolingual agents. Given the communicative and social functions and roles of CS, can CS be an effective strategy for dialog systems? Can the appropriate use of CS by a dialog system improve its task-effectiveness or human judgment of its responses?

Further, would users perceive CS agents as being more natural or engaging, or do the social norms around human interactions not shape expectations for human-agent conversations, as is suggested by Ciechanowski et al. (2018)? For many agent applications, there are no obvious advantages of introducing CS ability, as only fluent bilinguals can code-switch and therefore, should be capable of conversing in either of the languages. Even if there are tangible improvements in judgment, would they be limited to certain types of users, or certain interaction-contexts? What would be the influence of reciprocity, and the quality of CS in the generated responses?

In this paper, we address the effect of CS usage on user perceptions through an in-depth user-study on human-bot conversations. Users observe snippets of human-bot conversations and are asked to compare several bot variants for naturalness and relative preference. We observe that users are frequently polarized on their judgments of CS, and that improved quality of generated CS also improves user judgments significantly.

We also observe that reciprocative use of CS is judged more favorably, indicating that the sociolinguistic theory of interpersonal accommodation (Bawa et al., 2018) also extends to CS in human-bot interactions. We carefully design the experiment and presentation to isolate the effect of CS from the effects of all other properties of the con-

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 171–179

versations and their dialogs.

Our multifaceted analysis reveals several interesting facts, such as:

1. Code-switching is used/perceived as a linguistic style marker, and therefore its accommodation is judged positively, even in human-bot conversations.
2. Irrespective of whether users code-switch themselves, their attitude towards a chatbot that code-switches can be extremely positive or negative.
3. Among users with a positive attitude, judgments of CS chatbots positively correlate with the naturalness of CS utterances, and with the accommodativeness of the bot in terms of its CS usage.

Thus, several important repercussions are borne out of this study in the context of dialog system design for multilingual societies. As far as we know, this is the first study to discuss the notion of CS in human-bot conversations.

We also discuss sociological and computational considerations affecting design choices of conversational agents, and briefly propose a novel strategy for navigating multilingual interactions and estimating user attitudes.

The rest of the paper is organized as follows- we motivate the questions of interest from related work in Section 2, followed by the experiment itself in elaborate detail in Section 3. We then discuss the implications of the study and its results in Section 4, and conclude in Section 5.

2 Motivation and Related Work

A wide range of differing attitudes towards CS has been well documented (Dewaele and Wei, 2014) and clearly indicate that CS is a style marker in multilingual conversations. Dewaele and Wei (2013) show that not only sociolinguistic factors like age, gender, education and language proficiency, but also personality types of speakers (levels of emotional stability, tolerance to ambiguity, cognitive empathy and neuroticism) affect their attitude towards CS. While we show CS to be similar to other dimensions of linguistic style (Tausczik and Pennebaker, 2010) in its cohesive and accommodative characteristics in human conversations, it also differs from them in being a strong sociological indicator of identity (Auer, 2005). Because of this sociological dimension, users may have different attitudes towards CS, and these attitudes may vary with users' demographic profiles. We delineate such effects in the study.

A computational study of style accommodation (Danescu-Niculescu-Mizil et al., 2011) shows that style-accommodation is highly prevalent and exhibits great complexity in Twitter conversations.

Language interaction and socio-pragmatic utility of code-switching in multilingual societies is very well studied in linguistics (Scotton and Ury, 1977; Fishman, 1970; Ervin-Tripp and Reyes, 2005; Dewaele, 2010; Rudra et al., 2016). Due to the prevalence and naturalness of CS in human-human conversations, we argue that a CS agent can build better rapport with its user by connecting to their common multilingual identity. Further, certain pragmatic and socio-linguistic factors, such as formality of context (Fishman, 1970), age (Ervin-Tripp and Reyes, 2005), expression of emotion (Dewaele, 2010) and sentiment (Rudra et al., 2016), have been found to function as socio-pragmatic signals for language preference in CS conversations. Style convergence in a conversation signals warmth and reduced inter-personal distance (Myers-Scotton, 1995; Blom et al., 2000), and Bawa et al. (2018) have shown that choice of language (or code) exhibits interpersonal convergence or accommodation in human conversations. Therefore, users could expected to follow similar patterns of conversation with an agent, and could find responses that follow these patterns to be more natural, which makes a case for both a CS understanding and generation ability in conversational agents.

However, it is unclear how much of the human-human conversation norms reflect in, and have the same pragmatic effect in, human-agent conversations. While there is evidence that humans exhibit a "chameleon effect" when engaged in a social-interaction (Ward and Litman, 2007; Reitter et al., 2011), there is limited evidence of any convergence in human-agent interactions (Branigan et al., 2010).

We do not know of any human-agent study that explores the effects of multilingual usage and code-switching. Hill et al. (2015) show that there are significant linguistic differences between human-human and human-agent conversations, which might be due to users' adaptation to the limitations of the technology (Arif and Stuer-

zlinger, 2012; Vinciarelli et al., 2015). These observations suggest that in a multilingual society, users may effortlessly adapt to monolingual agents, just as they would to monolingual speakers. Our in-depth user study extends some of these findings to code-switching as a dimension of linguistic style, by measuring how code-choice and code-choice accommodation by conversational agents are perceived by different kinds of human users.

Though, as shown in another user study (Thies et al., 2017) on preferences over bot personalities that even a single user can have different preferences based on what they are trying to achieve, we might argue that depending on the context of usage of an agent, user might prefer a CS agent (for instance for personal assistant or chit-chat) or a monolingual agent (for a particular goal, like booking flight tickets, or finding scientific articles), and while multilingual users can easily adapt to a monolingual agent, a CS agent could still be perceived as more empathetic and engaging, providing a better overall user experience.

Yet another confounding factor is the wide range of differing attitudes towards CS that exist at the level of an individual or community of speakers (Dewaele and Wei, 2014). Dewaele and Wei (2013) show that not only sociolinguistic factors like age, gender, education and language proficiency, but also personality types of speakers (levels of emotional stability, tolerance to ambiguity, cognitive empathy and neuroticism) affect their attitude towards CS. Thus, the preference towards a CS agent might widely vary across individuals, communities and multilingual geographies.

Given the wide differences in the perception of CS, perception of chatbots, and the fact that insights from human-human conversations cannot be trivially applied to human-agent conversations, we cannot a priori comment on the usefulness of CS agents, and perform a user study to gain insights on these questions.

3 User Study on Human-Bot Conversations

The goal of the study is to quantify the causal effects of the following on the user judgments of agents:

1. Presence of CS in agent and human dialogs
2. Expressed attitude of users towards CS, as revealed and inferred from user comments
3. Naturalness and reciprocal nature of CS
4. Demographic profile of the users

Users are shown a series of human-bot conversation snippets in pairs, and asked to rate the agents (bots) on conversational skill and relative preference within the pair. The users are also asked to comment on the aspects of the conversations that they notice, the differences between the two agents' conversational skill, and the reasons for their stated preferences. These snippets average about 15 dialogs each in length, and vary in the presence of CS in dialogs by either dialog participant.

3.1 Experiment Design

There are four variants, or 'conditions' of each presented conversation : No CS by either participant ($None$), CS used by agent only ($Agent$), CS by human only ($Human$) and CS by both participants ($Both$). This is done to isolate the effect of *style* (here CS) from the effect of *content* of the conversations.

In each trial, a user is shown two human-agent conversation snippets, each with a different agent. Users are asked to rate each agent on perceived conversational skill on a 7-point Likert scale, ranging from 'Extremely Bad' to 'Extremely Good', and to compare the two agents by assigning relative preference between the two, again on a 7-point Likert scale, with 'Strong preference' for either agent as the extremes and 'No preference' as the mean.

In a pilot version, users were shown conversation pairs that differed only exactly in CS usage and were identical otherwise. This led to starkly negative judgments of CS, as CS usage stood out and was therefore judged critically. Since code-switching is a non-conscious process for many fluent bilinguals (Heredia and Altarriba, 2001), explicitly asking users to judge CS in this way is likely to distort judgments because of observer bias (Poulton, 1975).

Therefore, to mask the variables of interest, we conduct a single-blind study, and allow the human-agent conversations to differ not just in CS usage but in various other respects such as conversational topic, lengths of utterances, expressed sentiment and conveyed personality. We do not control for any variation across conversations, stylistic or otherwise, and treat all differences between conversa-

tions (except code-switching) as confounding covariates, or noise.

3.1.1 Presentation Order

Users are randomly divided into four groups; each user sees 4 pairs of conversations to be compared (so 8 conversations per user), where one was always in the $None$ condition, while the other varies across all four conditions mentioned above (including the $None$-$None$ pair, which provided the baseline judgments that we later use to measure the ratings for other conditions against). We employ a presentation order similar to the case-crossover study design (Lombardi, 2010), in which different randomly-assigned groups see different permutations of conversations and CS variants, as shown in Table 1.

Each group sees the same conversations (and in the same order) as other groups, with the corresponding conversations across the groups differing exactly in the CS conditions applied. Additionally, as the order of presentation of the CS variants varies across the groups, it takes care of ordering biases like fatigue and carryover (Strack, 1992).

	Conversation Pair			
Group	**1**	**2**	**3**	**4**
1	$None$	$Agent$	$Human$	$Both$
2	$Agent$	$Human$	$Both$	$None$
3	$Human$	$Both$	$None$	$Agent$
4	$Both$	$None$	$Agent$	$Human$

Table 1: Presentation order of the CS variants

3.1.2 Conversation Generation Methodology

The conversation snippets are extracted from real conversations between one of the authors and an online chatbot, Ruuh[1] that were conducted as a part of this study. We chose to use real conversations with a chatbot rather than artificially constructed chat so that the data reflects the nature and technological constraints of human-bot conversations.

Eight such conversation snippets were collected that span everyday topics such as hobbies, travel, relationships and work. A fluent English-Hindi bilingual was consulted to generate code-switched versions of the dialogs ($Agent$, $Human$ and $Both$ variants). In keeping with the findings on markedness of conversational English-Hindi CS from Bawa et al. (2018), the generated conversations have Hindi as the marked language and moderately high values of sentence-level fraction of Hindi words.

The box below shows one such conversation. The full list of questions and all conversations are provided in Supplementary Material.

Human	Do you play any musical instruments?
Bot	Nope, just listen to a lot of music I guess.
Human	*Main* piano play *karna sikh raha hoon* ⟨ I'm learning to play the piano ⟩
Bot	Piano *bajaanaa* is like a zen mood *mere liye*. ⟨ Playing the piano is like a zen mood for me ⟩
Human	Indeed, it is very relaxing to play the piano after a long day at work
Bot	Does learning how to play the piano help?
Human	*Isse* focus *karne me* help *hota hai*, and to develop a taste in music ⟨ It helps you to focus, and to develop a taste in music ⟩
Bot	Music *to ek* language *hai yaar*. You were born to speak it. ⟨ But music is a language dude. You were born to speak it. ⟩
Human	Yes, but some nuances need to be learnt.
Bot	No I do not think music can be learnt.

Sample conversation. Hindi-code is in *italics*.

3.2 Analysis

A total of 68 users participated in our study, all fluent English-Hindi bilinguals. A user typically took 15-20 minutes to answer all questions.

3.2.1 Demographic Profile

In addition to the judgment scores from users, we collected basic demographic information about each user- their age, native language and other languages known, places where they've lived in for at least three years, and highest level of education attained. Interestingly, none of these were found to be correlated with either attitude classes or with judgments. (exact values of these correlations are reported in Supplementary Material).

3.2.2 Comparing Judgment Ratings

The 7-point skill ratings are normalized to zero-mean and unit-variance for each user, as we are interested only in the *relative* ratings given to the CS variants by each user. The 7-point preference ratings over conversation pairs are similarly normalized to unit-variance.

Let $\textsc{Skill}_{V,C}$ denote this normalized skill rating for a conversation C presented with the CS variant V, as judged by all users who see this variant. Let $C1$ and $C2$ be the two conversations presented in some pair, with $C1$ presented

[1] https://www.facebook.com/Ruuh/

with CS variant V and $C2$ with the base variant ($None$). The corresponding relative skill rating of $C1$ over $C2$ is then defined as $\text{SKILL}_{V,C1,C2} = \text{SKILL}_{V,C1} - \text{SKILL}_{None,C2}$.

We adjust these ratings against the base difference between the conversations $C1$ and $C2$ (differences that can be attributed to everything except the code-switching) by getting $\text{SKILL}'_{V,C1,C2} = \text{SKILL}_{V,C1,C2} - \text{SKILL}_{None,C1,C2}$. We are able to do this because every conversation pair is also shown once to *some* group without any CS in either conversation ($None$ condition in Table 1). The overall skill rating of condition $V \in \{Agent, Human, Both\}$ is then just the average over all conversation pairs, $\text{SKILL}_V = \mathbb{E}_{C1,C2}(\text{SKILL}'_{V,C1,C2})$. Let $\text{PREF}_{V,C1,C2}$ denote the normalized preference rating of $C1$ over $C2$, as judged by all users who see this pair. We analogously derive the overall preference rating PREF_V.

3.2.3 Inferring Attitude Classes

We look at the text comments provided by users and classify users based on if they explicitly mention language mixing (or any paraphrasing of it) in one of the differences that they notice between the conversation pairs, and if they express any sentiment towards it. This gives us three types of users, differing in their expressed attitude towards CS:
Pos : CS is noticed; positive attitude or preference expressed. (39% users)
Neg : CS is noticed; negative attitude or dispreference expressed. (29% users)
$Neut$: CS is not mentioned or no sentiment can be discerned from comments. (32% users)

We do not observe any significant trends in values of SKILL_V and PREF_V across the population as a whole, but the distribution of these values does suggest some clustering of user ratings. Indeed, when we condition the ratings SKILL_V and PREF_V on the attitude class $A \in \{Pos, Neg, Neut\}$ of the users providing them, denoted by SKILL^A_V and PREF^A_V, we see significant differences.

3.2.4 Labeling CS Quality

We are interested in quantifying the effect of CS quality on user judgments. We had two independent annotators rate all conversations and variants on a 5-point scale for CS fluency and naturalness. We then binarize this judgment and label each conversation as having 'high-quality CS' or 'low-quality CS'. This divides the 4 conversations into two classes of two conversations each. Restricting SKILL_V only to conversations with CS quality $Q \in \{High, Low\}$, gives SKILL^Q_V.

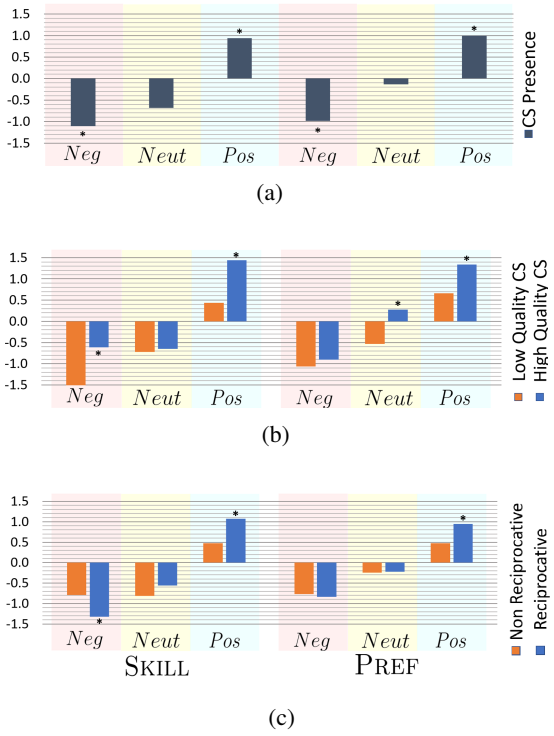


Figure 1: Skill and preference ratings across various conditions and user groups. (*) denotes significant difference from zero or between the pair.

3.3 Observations

Figure 1a shows $\text{SKILL}^A_{Bot+HumanBot}$ and $\text{PREF}^A_{Bot+HumanBot}$ for all A. They capture the effect of presence of CS in bot's dialogs. Figure 1b shows $\text{SKILL}^{Q,A}_{Bot+HumanBot}$ and $\text{PREF}^{Q,A}_{Bot+HumanBot}$ for all Q and A, and they show the effect of quality of CS of the bot's dialogs. Figure 1c) shows $\text{SKILL}^A_{HumanBot}$ against $\text{SKILL}^A_{Bot+Human}$ and $\text{PREF}^A_{HumanBot}$ against $\text{PREF}^A_{Bot+Human}$ for all A. This brings out the effect of accommodative CS on SKILL and PREF judgments.

3.3.1 Presence and Quality of CS

The effects of CS on these normalized judgments of skill (SKILL) and relative preference (PREF) for the users in the three attitude classes are seen in Figure 1. The effect of presence and quality of CS in agent dialogs are shown in Figure 1a and 1b respectively. For the attitude class Neg, all judgments of CS (irrespective of quality) are consistently negative, which is not surprising, as users

who have a dispreference for the phenomenon of CS itself are unlikely to have a notion of quality.

SKILL judgments by $Neut$ users are similar to those in Neg, but PREF judgments are sensitive to CS quality. We speculate that their dispreference can at least partially be addressed by improving CS generation by bots, or by aligning it better with known user CS patterns.

3.3.2 Reciprocity of CS

Figure 1c compares the ratings when only one of the parties code-switch ($Agent$ and $Human$) to when both code-switch in a reciprocative manner ($Both$). Pos users have a strong preference towards reciprocative CS, which is in line with reciprocity observed in human conversations (Bawa et al., 2018). This suggests that users in this class judge conversational agents on similar parameters as they would judge human interlocutors. Users in Pos not only rate CS highly, but are also sensitive to both quality and accommodativeness of bot's CS, with accommodative CS perceived as much more skillful than anti-accommodative CS.

3.4 Results

To summarize, the study points to two primary takeaways: (1) it is important to *know* or otherwise *identify* the "user's attitude" towards an agent that code-switches, as introducing CS has diametrically opposite effects on users with different attitudes, and (2) quality of CS responses is important to all users, and might also influence their attitude towards a CS-agent in the long run.

Overall, it is suggested from the study that good-quality accommodative CS significantly improves judgment for a large fraction of users whose general attitude towards CS has otherwise been identified.

Demographic factors are all poorly correlated with judgments of SKILL and PREF. Demographic variables also fail to predict attitude classes, with all classes having a similar spread of all demographic variables. See Supplementary Material for all such correlations.

4 Discussion

The findings of the study have multiple implications for the design of CS conversational agents, and their strategy for making CS decisions.

4.1 Attitude Estimation

The ability to infer a user's attitude towards CS seems to be the single-most important determinant of the success of any CS strategy by a bot, as a bot that can infer the attitude class can make an informed decision on whether or not to adopt CS. Users' attitude towards CS depends on various social and psychological variables (Dewaele and Wei, 2013). Since attitude estimation is crucial, it should be incorporated into the design of an agent. Individual disposition could be predicted at the demographic level, though we found no correlation between users' attitude and their demographics, suggesting that demography-based inference of attitude is unreliable.

Furthermore, it is not necessary that a person's attitude towards CS in human conversations matches that in a human-bot conversation. For instance, in our study, users commented: "I didn't like the way he was switching languages. That felt very forced." and "...is trying too hard to sound natural".

It seems straightforward to just ask the user about their CS preferences, but as CS choices could be nonconscious or spontaneous decisions like other aspects of linguistic style (Levelt and Kelter, 1982), stated preferences are unlikely to be as reliable as preferences revealed from observing users in-conversation (Levitt and List, 2007). Furthermore, the latter estimates would also reveal a user's individual style of CS and extent of CS.

4.2 Nudging as a Conversational Strategy

Such probing of CS preferences while conversing needs to be a balancing act. Without any adaptation, the agent is likely to stick to a suboptimal default. On the other hand, aggressive probing in the form of arbitrary CS decisions will immediately distract and annoy users.

We propose *nudging* as a strategy to navigate this tradeoff - the agent slightly deviates from its default response (in terms of CS) and measures if the change is reciprocated, either immediately or over a few turns, to implicitly estimate user preference. In principle, these changes can also be measured using other evaluation criteria (Liu et al., 2016). Nudging has been studied for human-agent interactions in other non-conversational contexts (Sadigh et al., 2016) and with the aim of influencing user behavior over digital interfaces (Weinmann et al., 2016). We propose nudging as an

active exploration strategy to reveal user preferences, over which longer term policies can then be studied. In our conversational context, 'nudging' would mean that the bot introduces instances of marked code-choice gradually in increasing amounts of markedness, while being sensitive to its measured effects on the user. These effects, in turn, could be inferred from their replies, from factors such as user's code-choice and/or trends in expressed sentiment.

4.3 CS Quality

We also see from the study that the perceived quality of generated CS matters, as bilinguals easily spot inaccuracies or unnatural CS patterns. Some of their responses within the user-study, to poor-quality CS, are: "answers in the style of speaker, all right, but the hinglish is unnatural", "... The place to change the language is a bit unnatural according to me" and "I don't like the way codemixing was used". Perceived CS quality affects judgments regardless of attitude.

Unless conversational systems can consistently generate high-quality CS, they may not be very well-received. CS quality itself depends on multiple factors. The first and most basic factor is syntactic soundness of mixed sentences. Joshi (1985) is the only work we know of that computationally generates grammatical CS sentences. The second determinant of perceived quality is the statistical likelihood of the CS pattern, which depends on the strength of the underlying language model. While there have been several proposed language models for CS (Adel et al., 2015; Ying and Fung, 2014), none of them have been evaluated against human judgments.

Quality judgments could be learned from natural CS data over Twitter, and there is initial work on making artificially generated CS similar to natural CS (Pratapa et al., 2018). Perceived CS quality, however, goes beyond the surface form of a sentence and is influenced by social and pragmatic functions in context (Begum et al., 2016).

While a better codification of naturalness judgments is needed, we speculate that an initial and safe strategy for an agent to explore nudging would be to introduce simpler CS constructions, like tags, frozen expressions (Poplack, 1988), or frequently observed discourse markers (Auer, 2013b). Monolingual responses from an existing conversational agent could be modified with such constructs in simple, even rule-based ways, to get corresponding CS versions.

5 Conclusion and Future Work

Our user study shows that proficient and accommodative CS in bots improves their perceived skill and preference for users who have a positive attitude towards CS by bots. In conclusion, we have argued that conversational agents need to discover and adapt to user CS preferences in order to gain relevance in multilingual contexts.

We propose nudging as a way to infer, on-the-fly, the users' attitude towards CS agents. In future work we would like to explore the utility of this strategy through a Wizard-of-Oz study, where the bot (wizard) can adaptively change the code-choice based on user's responses. Such strategies could also be automatically inferred from analysis of human-bot conversations in an inverse reinforcement learning framework as formulated in (Sadigh et al., 2016). Further, unlike our user study, a Wizard-of-Oz study will allow users to actively participate in the conversation and therefore provide better judgments.

References

Heike Adel, Ngoc Thang Vu, Katrin Kirchhoff, Dominic Telaar, and Tanja Schultz. 2015. Syntactic and semantic features for code-switching factored language models. *IEEE Transactions on Audio, Speech, and Language Processing*, 23(3):431–440.

Ana Inés Ansaldo, Karine Marcotte, Lilian Scherer, and Gaelle Raboyeau. 2008. Language therapy and bilingual aphasia: Clinical implications of psycholinguistic and neuroimaging research. *Journal of Neurolinguistics*, 21(6):539–557.

Ahmed Sabbir Arif and Wolfgang Stuerzlinger. 2012. How do users adapt to a faulty system. In *CHI'12 Workshop on Designing and Evaluating Text Entry Methods*, pages 11–14.

Peter Auer. 1995. The pragmatics of code-switching: A sequential approach. *One speaker, two languages: Cross-disciplinary perspectives on code-switching*, pages 115–135.

Peter Auer. 2005. A postscript: Code-switching and social identity. *Journal of pragmatics*, 37(3):403–410.

Peter Auer. 2013a. *Code-switching in conversation: Language, interaction and identity*. Routledge.

Peter Auer. 2013b. The'why'and'how'questions in the analysis of conversational code-switching. In *Code-switching in Conversation*, pages 164–187. Routledge.

Anshul Bawa, Monojit Choudhury, and Kalika Bali. 2018. Accommodation of conversational code choice. *Third Workshop on Computational Approaches to Linguistic Code-switching, ACL 2018*.

Rafiya Begum, Kalika Bali, Monojit Choudhury, Koustav Rudra, and Niloy Ganguly. 2016. Functions of code-switching in tweets: An annotation framework and some initial experiments. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA).

Jan-Petter Blom, John J Gumperz, et al. 2000. Social meaning in linguistic structure: Code-switching in norway. *The bilingualism reader*, pages 111–136.

Holly P Branigan, Martin J Pickering, Jamie Pearson, and Janet F McLean. 2010. Linguistic alignment between people and computers. *Journal of Pragmatics*, 42(9):2355–2368.

Antoinette Camilleri. 1996. Language values and identities: Code switching in secondary classrooms in malta. *Linguistics and education*, 8(1):85–103.

Leon Ciechanowski, Aleksandra Przegalinska, Mikolaj Magnuski, and Peter Gloor. 2018. In the shades of the uncanny valley: An experimental study of human–chatbot interaction. *Future Generation Computer Systems*.

Cristian Danescu-Niculescu-Mizil, Michael Gamon, and Susan Dumais. 2011. Mark my words!: linguistic style accommodation in social media. In *Proceedings of the 20th international conference on World wide web*, pages 745–754. ACM.

Anna De Fina. 2007. Code-switching and the construction of ethnic identity in a community of practice. *Language in society*, 36(3):371–392.

Jean-Marc Dewaele. 2010. *Emotions in multiple languages*. Palgrave Macmillan, Basingstoke, UK.

Jean-Marc Dewaele and Li Wei. 2013. Is multilingualism linked to a higher tolerance of ambiguity? *Bilingualism: Language and Cognition*, 16(1):231–240.

Jean-Marc Dewaele and Li Wei. 2014. Attitudes towards code-switching among adult mono-and multilingual language users. *Journal of Multilingual and Multicultural Development*, 35(3):235–251.

Susan Ervin-Tripp and Iliana Reyes. 2005. Child codeswitching and adult content contrasts. *International Journal of Bilingualism*, 9(1):85–102.

J.A. Fishman. 1970. *Sociolinguistics: a brief introduction*. Newbury House language series. Newbury House.

John J Gumperz. 1982. *Discourse strategies*, volume 1. Cambridge University Press.

Roberto R Heredia and Jeanette Altarriba. 2001. Bilingual language mixing: Why do bilinguals code-switch? *Current Directions in Psychological Science*, 10(5):164–168.

Jennifer Hill, W Randolph Ford, and Ingrid G Farreras. 2015. Real conversations with artificial intelligence: A comparison between human–human online conversations and human–chatbot conversations. *Computers in Human Behavior*, 49:245–250.

A. K. Joshi. 1985. Processing of Sentences with Intrasentential Code Switching. In D. R. Dowty, L. Karttunen, and A. M. Zwicky, editors, *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives*, pages 190–205. Cambridge University Press, Cambridge.

Willem JM Levelt and Stephanie Kelter. 1982. Surface form and memory in question answering. *Cognitive psychology*, 14(1):78–106.

Steven D Levitt and John A List. 2007. What do laboratory experiments measuring social preferences reveal about the real world? *Journal of Economic perspectives*, 21(2):153–174.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

David A Lombardi. 2010. The case-crossover study: a novel design in evaluating transient fatigue as a risk factor for road traffic accidents. *Sleep*, 33(3):283–284.

Carol Myers-Scotton. 1993. *Dueling Languages: Grammatical Structure in Code-Switching*. Claredon, Oxford.

Carol Myers-Scotton. 1995. *Social motivations for codeswitching: Evidence from Africa*. Oxford University Press.

Mark Myslín and Roger Levy. 2015. Code-switching and predictability of meaning in discourse. *Language*, 91(4):871–905.

Shana Poplack. 1988. Contrasting patterns of code-switching in two communities. *Codeswitching: Anthropological and sociolinguistic perspectives*, 215:44.

EC Poulton. 1975. Observer bias. *Applied ergonomics*, 6(1):3–8.

Adithya Pratapa, Gayatri Bhat, Monojit Choudhury, Sunayana Sitaram, Sandipan Dandapat, and Kalika Bali. 2018. Language modeling for code-mixing: The role of linguistic theory based synthetic data. In *ACL*.

David Reitter, Frank Keller, and Johanna D Moore. 2011. A computational cognitive model of syntactic priming. *Cognitive science*, 35(4):587–637.

Koustav Rudra, Shruti Rijhwani, Rafiya Begum, Kalika Bali, Monojit Choudhury, and Niloy Ganguly. 2016. Understanding language preference for expression of opinion and sentiment: What do Hindi-English speakers do on Twitter? In *EMNLP*, pages 1131–1141.

Dorsa Sadigh, S Shankar Sastry, Sanjit A Seshia, and Anca Dragan. 2016. Information gathering actions over human internal state. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 66–73. IEEE.

Carol Myers Scotton and William Ury. 1977. Bilingual strategies: The social functions of code-switching. *International Journal of the sociology of language*, 1977(13):5–20.

Heung-Yeung Shum, Xiaodong He, and Di Li. 2018. From eliza to xiaoice: Challenges and opportunities with social chatbots. *arXiv preprint arXiv:1801.01957*.

Fritz Strack. 1992. order effects in survey research: Activation and information functions of preceding questions. In *Context effects in social and psychological research*, pages 23–34. Springer.

Yla R Tausczik and James W Pennebaker. 2010. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of language and social psychology*, 29(1):24–54.

Indrani Medhi Thies, Nandita Menon, Sneha Magapu, Manisha Subramony, and Jacki Oneill. 2017. How do you want your chatbot? an exploratory wizard-of-oz study with young, urban indians. In *IFIP Conference on Human-Computer Interaction*, pages 441–459. Springer.

Alessandro Vinciarelli, Anna Esposito, Elisabeth André, Francesca Bonin, Mohamed Chetouani, Jeffrey F Cohn, Marco Cristani, Ferdinand Fuhrmann, Elmer Gilmartin, Zakia Hammal, et al. 2015. Open challenges in modelling, analysis and synthesis of human behaviour in human–human and human–machine interactions. *Cognitive Computation*, 7(4):397–413.

Arthur Ward and Diane Litman. 2007. Dialog convergence and learning. *Frontiers in Artificial Intelligence and Applications*, 158:262.

Markus Weinmann, Christoph Schneider, and Jan vom Brocke. 2016. Digital nudging. *Business & Information Systems Engineering*, 58(6):433–436.

Li Ying and Pascale Fung. 2014. Language modeling with functional head constraint for code switching speech recognition. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 907–916.

Hierarchical Text Generation using an Outline

Mehdi Drissi
Harvey Mudd College
Claremont, CA, USA
mdrissi@hmc.edu

Olivia Watkins
Harvey Mudd College
Claremont, CA, USA
owatkins@hmc.edu

Jugal Kalita
University of Colorado Springs
Colorado Springs, CO, USA
jkalita@uccs.edu

Abstract

Many challenges in natural language processing require generating text, including language translation, dialogue generation, and speech recognition. For all of these problems, text generation becomes more difficult as the text becomes longer. Current language models often struggle to keep track of coherence for long pieces of text. Here, we attempt to have the model construct and use an outline of the text it generates to keep it focused. We find that the usage of an outline improves perplexity. We do not find that using the outline improves human evaluation over a simpler baseline, revealing a discrepancy in perplexity and human perception. Similarly, hierarchical generation is not found to improve human evaluation scores.

1 Introduction

Recurrent neural networks have been successfully used for a variety of tasks in dealing with natural language. Successes include language translation (Sutskever et al., 2014), speech recognition (Graves, 2012), and text to speech (Kalchbrenner et al., 2018). They all learn to model the conditional probability of a sequence and generate words sequentially, typically using sequence to sequence models. These models have the advantage that their negative log likelihood is differentiable, allowing them to be directly trained through gradient descent.

A similar task is language modeling. Here the goal is to determine the probability of a sequence of words. Being able to model text is important for natural language understanding. These models can be used for re-ranking candidate translations (Cho et al., 2014), determining possible ways to extend a piece of writing (Roemmele and Gordon, 2018), and generating text (Dauphin et al., 2016). These problems all share the difficulty that although in theory a recurrent model can preserve information for arbitrarily long sequences, in practice recurrent models tend to struggle to keep track of context as the sequence length becomes large. Models for tasks like language translation tend to avoid translating entire paragraphs and instead focus on only generating a sentence at a time. Similarly, when one generates multiple sentences of text from language models, they tend to be locally coherent but not globally coherent.

The difficulty of generating large samples is not unique to text and also occurs with images. In the realm of images though, a different technique is commonly used for generation. Here, generative adversarial networks (Goodfellow et al., 2014) have been successfully used to generate images directly. Similar to text, initially these models only worked well for generating small images. Generating large images (like 1024 by 1024) was difficult for these models. In some recent work, the issue of large images was dealt with by generating images in a hierarchical manner. Instead of directly learning to generate the desired image, lower resolution images were first generated and then improved upon (Zhang et al., 2016). Initially, this was done by generating one lower resolution image and then directly the final image. More recently, this has been extended to starting off with generating a small image and iteratively increasing its resolution by double until reaching the desired size (Karras et al., 2017).

Inspired by the idea of generating images in a hierarchical manner, here we will explore generating text in a hierarchical manner. Similar to (Zhang et al., 2016), we will approach this by generating in two steps. One difficulty with hierarchical text generation is that meaningfully down-

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 180–187

sampling text is more difficult than downsampling images. The textual equivalent of decreasing resolution is summarization, which is a difficult problem in itself. To side-step this issue, we will use a simple extractive summarization approach to get an outline. Using an extractive summarization approach to acquire information to build upon has been done previously in generating Wikipedia articles (Liu et al., 2018). Their approach differs from ours in that there they used summarization to extract relevant information from references to generate the article, while here we are summarizing the target text to acquire an outline of it. We generate complete documents hierarchically with two models. First, one model component generates the outline. We then condition upon the outline to generate the entire document.

Our main contribution will be to explore generating text in a hierarchical manner by separating text generation into two phases. The first phase generates an outline of the text, while the second phase uses the outline to generate the complete article.

2 Background

Most sequence-to-sequence models consist of an encoder and decoder model (Sutskever et al., 2014). The encoder model's purpose is to take in an input sequence and construct representations of each token in the sequence. The decoder's hidden state is initialized with encoder's final hidden state. At each step, the decoder predicts the next token in the sequence until it predicts an end of sequence token.

As it is difficult to encode an entire sequence into one vector, generally the decoder is allowed to look back at the representations of the tokens the encoder created through an attention mechanism (Bahdanau et al., 2014). An attention mechanism scores the decoder's hidden state against the encodings of all the tokens in the input sentence, converts these scores to probabilities, and then takes a weighted average of each encoding to produce the context vector. This context vector is then fed in to the decoder to aid it in keeping track of information from the entire sentence.

One weakness of conventional attention is that it can only focus on words in the prompt sequence and not in the target sequence. Self-attention (Vaswani et al., 2017) is a modification that attends to both words in the prompt and prior words in the target. This is especially important for models that generate long sequences to be able to keep track of what has been made. A second weakness of conventional attention when used on long sequences of input text is that it only operates at the word level, ignoring any information conveyed through the fact that words are grouped into sentences. An extension of attention to account for both word level and sentence level information is hierarchical attention (Ling and Rush, 2017), which constructs and scores a vector for each sentence along with each word. The probability of focusing on a word is then the product of its associated word level attention and sentence level attention.

For summarization, approaches fall into two primary categories. Extractive summarization involves choosing and directly copying the main sentences/words from the document. Abstractive summarization focuses on having a model generate the words directly for the summary. As our goal is to simply have an outline of the text, extractive summarization is sufficient. It would be interesting future work to see how different methods of generating an outline affect document generation. Another possible future extension would be to instead use topic modeling to extract a different form of useful context for the target text.

The summarization algorithm that will be used is called SumBasic (Nenkova and Vanderwende, 2005). This algorithm is based upon choosing sentences with words that are frequent in the document. After choosing a sentence, the algorithm down-weights the words in that sentence to avoid choosing sentences that are too similar.

3 Methods

Our goal is to generate text in a hierarchical manner by generating the topic sentences of the article first and then generating the entire paragraph by conditioning on the topic sentence.

3.1 Model Overview

The model will be an extension of the model used in Neural Story Generation (Fan et al., 2018). In that work, a sequence-to-sequence model was used to convert a prompt into a complete document. Our main extension is dividing the sequence-to-sequence model into two components to assist in the coherence for the generated text. One component will generate topic sentences, and one will expand topic sentences into

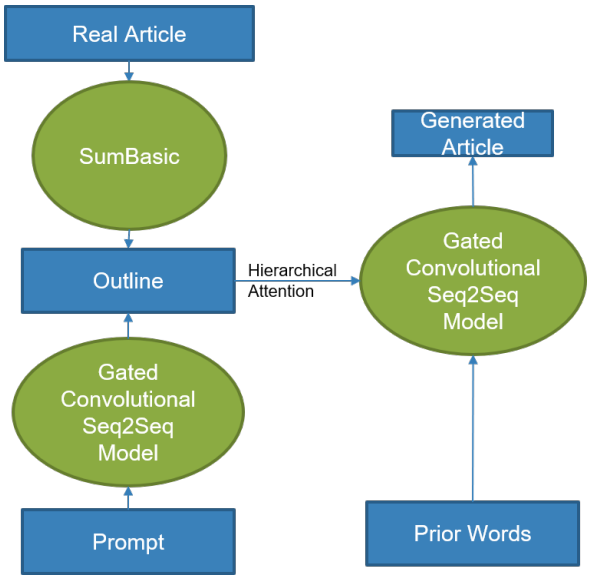


Figure 1: This diagram gives a high level overview of the article generation process. Outlines are prepared with SumBasic. The prompt is converted to an outline which is then converted to an article.

complete paragraphs.

3.2 First Component - Outline Generator

To generate a document from the prompt, we first generate the outline associated with the document. This component will be trained using almost an architecture almost identical to the model used by (Fan et al., 2018). Their architecture is a convolutional sequence-to-sequence model (Gehring et al., 2017) that uses a gated form of self-attention. The one difference is we did not use the cold fusion (Sriram et al., 2017) mechanism, mainly to lower training time. Cold fusion is a mechanism that involves first training the main model, freezing its weights, and then training a second randomly initialized version of the main model, concatenating the two models' outputs, and then adding a few layers to determine the final prediction.

3.3 Second Component - Article Generator

The second component will use the outline to generate the entire article. It will also be based upon the sequence-to-sequence model used by (Fan et al., 2018) and similarly will not include cold fusion. It will be extended in one way. The self-attention heads in the decoder will remain the same, but the attention over the encoded outline will be replaced by hierarchical attention. The sentence vectors will be obtained by summing the encoded word vectors for each word in the corresponding sentence. The sentence vectors will then also have the same type of attention mechanism applied to them. Similar to how the prior decoder attention was gated, the hierarchical attention is also gated and uses the same gating. Both model components will be trained separately by directly optimizing the negative log likelihood.

3.4 Outlines

To train the model, we require a dataset of outlines corresponding to complete articles. As the primary focus of this work is not on new methods of summarization, these outlines will be generated using prior text summarization methods. SumBasic (Nenkova and Vanderwende, 2005) is one frequency-based method for determining the topic sentence. It tries to find sentences whose words are common in the document. One way to avoid overweighting common words like 'the' is to penalize words that are common across many articles. TF-IDF (Term Frequency Inverse Document Frequency) does this by multiplying by the negative log probability of a word appearing in a document. SumBasic is often tweaked to use TF-IDF instead of directly using word frequencies (Allahyari et al., 2017).

As the outline is intended to contain information about each section of the text, instead of directly applying SumBasic to the full document, it will be applied at the meta-paragraph level. Here, meta-paragraph does not refer to the actual paragraphs in the text, because their lengths are very inconsistent. The paragraphs of many of the documents in the training data only consist of one or two sentences. Extracting a topic sentence from each of these paragraphs as an outline would be problematic as we would effectively be letting the outline contain too much of the document. To avoid this issue, the actual paragraphs will be aggregated together using the rule that any paragraph under a threshold number of sentences k will be combined with the next paragraph to form a meta-paragraph. As a side effect, every meta-paragraph except for possibly the last one will end up having at least k sentences.

Lastly, some preprocessing is done before directly applying SumBasic. Stop words are removed using a list of NLTK's (Loper and Bird, 2002) English stop words, numbers are removed,

punctuation is removed, and words are stemmed using Porter stemming (Porter, 1980).

3.5 Training the Models

Strictly speaking the probability of an entire document can only now be obtained by marginalizing over all possible outlines. A lower bound for the probability can be obtained by instead generating the most likely outline for a given prompt and then finding the probability of the document conditioned on that outline. This lower bound however is intractable to compute as it involves generating many outlines. Due to the long length of the sequences and the relatively slow generation time, even only using 10 outlines for the approximation would take approximately 100 days just to evaluate on the validation/test set on one 1080 Ti. It also turns out to be intractable as the memory needed to do a beam search with such long sequences is too high and it runs out of memory if you increase the beam size beyond 2.

As the two components are trained separately, this similarly leads to the overall training loss not corresponding to the actual article negative log likelihood. Using that loss properly would require being able to efficiently compute the probability of a document, but this is intractable for the reasons given in the prior paragraph.

A second discrepancy that arises is that since the components are trained separately, the second component is only trained on good outlines. It's never trained to deal with poorly generated outlines, so if the first component generates a bad outline, the second component is unlikely to be able to recover from the mistake. This is unlike the training for StackGAN (Zhang et al., 2016) where the second part of the model was trained using the output of the first part of the model. Training this model in an end-to-end manner is difficult because the slow outline generation time would increase the training time by a factor of roughly 30 which was unfeasible given our computational resources. Training on a small amount of generated samples to fine-tune the model is feasible and could be done in future work.

3.6 Code

A github repository containing our model code can be found here: https://github.com/hmc-cs-mdrissi/fairseq. The dataset of prompts, articles, and outlines can be found at this link: https://www.dropbox.com/s/jupoljc2cx0to7y/datasets.zip?dl=0.

4 Evaluation Approach

4.1 Datasets

We used the Wikitext-103 dataset described in (Merity et al., 2016). It consists of a large collection of preprocessed Wikipedia articles. As this dataset does not come with prompts, the prompt used was the first sentence of the article, with the goal being generating the entire article. We preprocessed the dataset to eliminate tables, to canonicalize numbers, and to lowercase each word.

4.2 Possible Evaluation Metrics

The two automated evaluation metrics used previously by (Fan et al., 2018) were perplexity and prompt relevance. BLEU and ROUGE were not used as the primary goal is generating new, diverse text rather than matching the target text precisely. Perplexity and prompt relevance are problematic for the hierarchical model as they both involve computing the probability of a article. Due to the previously discussed computational intractability of computing article probabilities, neither can be used for the hierarchical model. For the non-hierarchical models and the components of the hierarchical model, we can still measure the perplexity to see how well the model captures the text distribution. Perplexity is the exponential of the cross entropy of the model distribution compared to the data distribution and is defined in Equation 1, where $q(x)$ is the probability the model assigns to x.

$$perplexity = 2^{-\frac{1}{N}\sum_{i=1}^{N}\log_2 q(x_i)} \quad (1)$$

Recently, (Semeniuta et al., 2018) explored various evaluation metrics for language models. One automated metric they explored was the Frechet Infersent Distance (FID). The FID metric is computed by encoding all of the generated and real documents as vectors and then comparing the distributions of these vectors by approximating them as Gaussian. The FID metric is problematic as it requires a model that can encode the meaning of a document well. In their work, they only focused on sentence-level generation and were able to use a model that could create sentence vectors. A second issue with the FID metric is that they did not find it sensitive to word order, which indicates that

Table 1: Model Perplexity

Model	Validation Perplexity
prompt-to-outline	45.63
outline-to-article	21.08
outline-to-article + h.a.	20.49
prompt-to-article	30.96

the metric does not correlate well with human perception of quality.

One, method applicable to any model is human evaluation. We generated about 27 articles for each model using the first sentences of articles in our test set as prompts. We then had native English speakers evaluate ten articles on both global coherence and overall quality on a 5 point Likert Scale. The questions that were used are shown in Figure 2.

4.3 Models

The models that will be compared are a model from prompt-to-outline, outline-to-article, outline-to-article + hierarchical attention (h.a.), prompt-to-article, hierarchical prompt-to-article + h.a..

5 Evaluation Results

5.1 Perplexity Evaluation

The model perplexity experiments revealed multiple interesting findings. The biggest one is that the perplexity of prompt-to-article is lower than the perplexity of prompt-to-outline, indicating that it is easier on average word-wise to generate the full article than just the outline. One possible explanation for this is that in generating the outline there are much more abrupt shifts in topic when compared to generating the article, and as each word is conditional on the prior words, the topic flowing more smoothly may make it easier to guess the next word. There is a significant improvement in perplexity (about 10 points) when conditioning on outlines compared to conditioning on prompts.

The other interesting result is that hierarchical attention led to a small improvement in perplexity. In the prior work by (Ling and Rush, 2017), hierarchical attention improved the model by providing a computationally efficient attention mechanism over long sequences and was not found to be helpful otherwise. A similar magnitude improvement using hierarchical attention occurred when comparing the two models on the stories dataset used by (Fan et al., 2018). Two plausible explanations are either that hierarchical attention works better for some tasks/datasets, or that the discrepancy occurred because we used gated attention for both the word-level attention model and the hierarchical attention model. The gating may have helped the sentence attention learn to focus on different and useful things than the word attention. Lastly, the hierarchical model is missing from the table as computing perplexity for it was intractable.

5.2 Human Evaluation

The results can be found in Table 5.1 on the next page. The number of articles evaluated total is 70. For the three models, only one pair is significantly different in global coherence. That pair is hierarchical model vs prompt-to-article where the latter is better ($p = 0.005$). The hierarchical model is also significantly different in quality from both of the other two models (for both $p = 0.049$). The worst model is still within a standard deviation evaluation wise of the best model in both metrics, indicating how noisy the quality of the articles and the reviews of them are. Thus, for human perception all the hierarchical model performed slightly worse, while the outline vs prompt models performed equally well.

The biggest unexpected result is that the prompt-to-article model performed better in human evaluation on global coherence than the outline-to-article model. Considering the outline describes the article more thoroughly, it should have helped the model stay focused. It was also unexpected that the overall quality of the outline-to-article would be the same as the prompt-to-article model. Both of these results, are unexpected because there was a large improvement in perplexity when conditioning on an outline. This reveals that for this task, perplexity did not correlate well with human perception of quality. Lastly, overall quality and global coherence correlate strongly ($r = 0.74$, $p < 0.001$).

5.3 Best Cases

Due to space constraints, articles and outlines were both truncated. unk was used for unknown tokens, num was used for numbers, and newline was used to keep track of paragraph breaks. The article shown in Figure 3 came from the outline-to-article model and tied for highest human evaluation scores.

Question 1 (Global Coherence) The article should be well-structured and well-organized. The article should not just be a heap of unrelated information, but should build from sentence to sentence and paragraph to paragraph. Abrupt changes in topic without any transitions are problematic.
Question 2 (Overall Quality) How realistic is the article as a whole given that the article is meant to be a Wikipedia article.

Figure 2: Questions for Human Evaluation of Generated Articles

Model	μ-Global Coherence	σ-Global Coherence	μ-Quality	σ-Quality
prompt-to-article	3.36	1.00	2.91	1.07
hierarchical prompt-to-article + h.a.	2.54	0.96	2.26	0.89
outline-to-article + h.a.	3.14	1.31	2.90	1.22

Table 2: The μ is for mean and σ is for standard deviation.

In both cases we see that the text conditioned upon was very informative. Doing a better job of preprocessing would likely help in some cases, as some of the other prompts end up being truncated too heavily (mainly due to difficulty determining where a sentence ends). While the main topic is preserved, often the facts will end up being changed. For the second article, originally the player was a basketball player, but the model morphed him into a baseball player.

5.4 Failure Cases

Here, we will examine one of the worst generated articles. Worst is defined as being rated as a 1 in both global coherence and overall quality. The article, shown in Figure 4 was generated from the hierarchical model.

This example looks to be a case where the first component of the hierarchical model did poorly leading to the model struggling to generate a article from that outline. It is not the only example where too many unknown word tokens end up leading to poor articles. While repetition occurs more broadly, unknown tokens have the worst tendency to lead the article to become too repetitive. This issue may be helped by training the second component of the hierarchical model on not just good outlines, but also sampled outlines so it can learn to not be too reliant on the outline.

More broadly, the main type of error found in generated articles is too much redundancy. Often individual sentences would not be too bad, but a very similar sentence would appear a bit later in the paragraph. Article quality also generally becomes worse near the end of the article.

6 Conclusion

We have focused on generating text in a hierarchical manner for generating Wikipedia articles. The primary automated metric, perplexity, for measuring overall article generation quality was computationally intractable for the hierarchical model. We found that conditioning on an outline heavily improved perplexity, but did not improve human perceived quality. This indicates that, perplexity should be used more cautiously for text generation if the motivation is generating text people will find believable. This issue should be analyzed more thoroughly in future work as perplexity is the most commonly used evaluation metric for language models currently. More generally, it would be useful to develop automated metrics that more closely correlate with human evaluation and have a better understanding of what qualitative aspects good articles share.

Other interesting findings are that that one major failure mode for the hierarchical model is difficulty dealing with poor outlines and that hierarchical attention led to a small improvement in perplexity for two different datasets unlike when it was used previously by (Ling and Rush, 2017).

For future work, the main thing is to deal with the issue of the hierarchical model not being able to generate reasonable articles with poor outlines. Training using only generated articles would be ideal, but is computationally expensive due to generation time being hundreds of times slower than training time. To make training feasible with generated samples, it will likely require only using a small number of generated articles to help fine tune the model. This may also be feasible when

Outline 1: he was previously the head coach at butler university in indianapolis . newline on april num , num , he became the head coach after todd unk left to coach the iowa hawkeyes . newline in july num , he signed a six - year , num million dollar contract to become the head coach of the boston celtics in the nba . newline bradley kent stevens grew up in unk , indiana , where he developed his love for basketball . newline he wore no. num in high school in honor of indiana pacers guard reggie miller . newline he earned three letters in basketball , three in track , and one in baseball during his days at unk . newline the more success you had at lilly , the harder it would be to leave . ”
Generated Article 1: walter george stevens (may num , num october num , num) was an american major league baseball (mlb) outfielder who played for the boston red sox (num num , num num) and the yankees (num num) . stevens was an outfielder , placekicker and punter , and had played with the boston indians and boston red sox from num to num . he was previously the head coach at butler university in indianapolis . he played num seasons in major league baseball (mlb) for the boston red sox , and in num played on the chicago red sox from num to num . newline stevens played num major league baseball (mlb) all - star team in num , and led the league to five consecutive national champions with a num num conference record , the longest streak in career history . he was also the only player to reach the major division i ’s num playoffs . the num and num seasons have taken place in the central division (num num) . newline at the time of the num mlb draft , he was known for the right to handle a hit . he was one to only two players . in num , the red sox traded him in minor league organization with the red sox . the red sox promoted him with the numth overall pick in the num mlb draft , while the red sox traded him to the cleveland indians .

Figure 3: Well-generated Article

the training dataset size is small.

Another possible extension is to make better usage of the outlines. Specifically, for ideal outlines we know that the sentences present in the outline should also be present in the target text. Allowing the model to copy an entire sentence would be beneficial and make it more likely to fully utilize the outline. This does come with the downside that a poor sentence in a generated outline being copied would be problematic. That could be dealt with by training on a mixture of generated and real outlines.

7 Acknowledgements

I would like to thank the UCCS NLP lab for many helpful conversations. This material is based upon work supported by the National Science Foundation under Grant No. 1659788. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. 2017. Text Summarization Techniques: A Brief Survey. *International Journal of Advanced Computer Science and Applications*, (1).

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*, abs/1409.0473.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language Modeling with Gated Convolutional Networks.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical Neural Story Generation. *ArXiv e-prints*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. *International Conference on Machine Learning*.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. pages 1–9.

Alex Graves. 2012. Sequence Transduction with Recurrent Neural Networks. *International Conference of Machine Learning*.

Prompt 1: the portage to san cristobal of a.h. is a num literary and philosophical novella by george steiner , in which jewish nazi hunters find a fictional adolf hitler (a.h.) alive in the amazon jungle thirty years after the end of world war ii .
Generated Outline 1: he began work as a unk at the age of num . newline during world war ii , unk joined the unk unk (royal society) in the united states where he made his world debut in the world . newline unk 's first name was unk unk . newline he was promoted to unk 's unk unk - unk in num . newline unk became the new man 's second - oldest student , unk unk - unk , as a student from unk - unk . newline unk was one of the founding members of the unk unk - unk , the first to be called unk .
Generated Article 1: unk unk (unk , " young woman 's sister , the unk , " to the unk ") is unk (unk , " young woman 's sister , the unk) , " girl 's son , and the unk " . the name unk is from the greek unk or " woman 's mother unk " , referring to her marriage to unk . after the marriage , her son , ila , is the only person to be killed in the world . newline during world war ii , il began work as a unk at the age of num . she was the first female woman to be a unk (unk) . after being married , ila became the new man 's second - oldest student ; however , her marriage to unk was interrupted in num . newline during world war ii , il joined the unk unk (royal society) in the united states where he made his world debut in the world . the unk - unk unk (unk) , the first of the first unk (unk) , was the first unk to be a unk , the second unk in the world , and a member of the unk unk , one of the first women 's women 's women 's women and is the first female woman to be a unk . in her unk unk , ila was first unk unk unk ; the other two are the unk (unk) , the unk (unk) , the (unk) , the unk (unk) , the unk (unk) , the unk (unk) , the unk (unk) , the unk (unk) and the (unk) . ila 's first name was unk unk . she was also used to refer to her .

Figure 4: A Poorly Generated Article

Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. 2018. Efficient Neural Audio Synthesis.

Tero Karras, Tima Aila, Samuli Laine, and Jaakko Lehtinen. 2017. Progressive Growing of GANs for Improved Quality, Stability, and Variation. *International Conference on Learning Representations*, pages 1–25.

Jeffrey Ling and Alexander M. Rush. 2017. Coarse-to-Fine Attention Models for Document Summarization. *Proceedings of the Workshop on New Frontiers in Summarization*, (2015):33–42.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating Wikipedia by Summarizing Long Sequences. pages 1–18.

Edward Loper and Steven Bird. 2002. NLTK: the natural language toolkit. *Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, cs.CL/0205028.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer Sentinel Mixture Models.

Ani Nenkova and Lucy Vanderwende. 2005. The impact of frequency on summarization. *Technical Report, Microsoft Research*.

Martin F Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Melissa Roemmele and Andrew S Gordon. 2018. Linguistic Features of Helpfulness in Automated Support for Creative Writing. *Naacl*, pages 14–19.

Stanislau Semeniuta, Aliaksei Severyn, and Sylvain Gelly. 2018. On Accurate Evaluation of GANs for Language Generation.

Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. 2017. Cold Fusion: Training Seq2Seq Models Together with Language Models.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. (Nips).

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. 2016. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. *International Conference on Computer Vision*.

SMT vs NMT: A Comparison over Hindi & Bengali Simple Sentences

Sainik Kumar Mahata[1], Soumil Mandal[2], Dipankar Das[3], Sivaji Bandyopadhyay[4]
[134]Jadavpur University, Kolkata
[2]SRM University, Chennai
[1]sainik.mahata@gmail.com, [2]soumil.mandal@gmail.com,
[3]dipankar.dipnil2005@gmail.com, [4]sivaji_cse_ju@yahoo.com

Abstract

In this article, we identified the qualitative differences between Statistical Machine Translation (SMT) and Neural Machine Translation (NMT) outputs. We have tried to answer two important questions: **1.** Does NMT perform equivalently well with respect to SMT and **2.** Does using simple sentences as training units, add extra flavor in improving the quality of Machine Translation output. In order to obtain insights, we have developed three core models viz., SMT model based on Moses toolkit, followed by character and word level NMT models. All of the systems use English-Hindi and English-Bengali language pairs containing simple sentences as well as sentences of other complexity. In order to preserve the translations semantics with respect to the target words of a sentence, we have employed soft-attention into our word level NMT model. We have further evaluated all the systems with respect to the scenarios where they succeed and fail. Finally, the quality of translation has been validated using BLEU and TER metrics along with manual parameters like fluency, adequacy etc. We observed that NMT outperforms SMT in case of simple sentences whereas SMT outperforms in case of all sentence types.

1 Introduction

Machine Translation (MT) refers to automated translation. It is the process by which computer software is used to translate a text from one natural language (such as English) into another (such as Spanish). Translation itself is a challenging task for humans, and hence, is more challenging for computers. High quality translation requires a thorough understanding of syntax and semantical properties of both the source and target languages.

The importance of studying and developing better MT systems has gained popularity in the recent past due to rapid globalization, where people from multiple backgrounds having varying language knowledge are working together. Primarily two paradigms are currently followed for building MT systems. One is based on statistical techniques, while the other employs artificial neural networks.

The statistical model, commonly referred to as Statistical Machine Translation (SMT) (Weaver, 1955), addresses this challenge by creating statistical models, whose input parameters are derived from the analysis of parallel bilingual text corpora (Mahata et al., 2017). Some of the notable works on SMT are (Al-Onaizan et al., 1999; Lopez, 2008; Koehn, 2009), where the authors have dived deep into the challenges, working principles and possible improvements. SMT has shown good results for many language pairs and is responsible for the recent surge in the popularity of MT among general public .

On the other hand, despite being relatively new, Neural Machine Translation (NMT) (Bahdanau et al., 2014) has already shown promising results (Mahata et al., 2016; Wu et al., 2016) and hence has gained substantial attention and interest. Continuous recurrent models for translation, which do not depend on alignment or phrasal translation units, was introduced by Kalchbrenner and Blunsom (2013). The problem of rare word occurrence was addressed by Luong et al. (2014) and the effectiveness of global and local approach was explored by Luong et al. (2015). He et al. (2016) demonstrated a log-linear framework incorporating SMT features combined with NMT which ad-

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 188–195

dresses out of vocabulary and inadequate translation. The properties of these architecture was discussed in detail in (Cho et al., 2014). This approach generally creates much more accurate translation than SMT given sufficient amount of training data (Vaswani et al., 2013; Liu et al., 2014; Doherty et al., 2010).

In the current work, we have tested the performance of SMT and NMT on simple sentences (see Section 2) extracted from English-Hindi (En-Hn) and English-Bengali (En-Bn) parallel corpus provided by TDIL[1]. These experiments were done to dive into the scenarios where NMT and SMT outperform each other. Moreover, they would also help us in evaluating the question that whether usage of simple sentences, when training the MT models, creates any difference in the quality of the MT output.

We have constrained our language domain to Hindi and Bengali as these languages are used primarily in the Indian sub-continent. Number of native speakers of Hindi in India is 41.1% while that of Bengali is 8.11%. Hindi is written in Devanagari [2] script and Bengali is written in Eastern Nagari [3] script.

To test the effectiveness of the case study, SMT and NMT systems were also trained for the whole corpus, which consists of sentences with mixed complexity. For both simple sentence corpus and the whole corpus, BLEU (Papineni et al., 2002), TER (Snover et al., 2006) and manual evaluation metrics like fluency and adequecy were calculated to validate the observed results.

The paper has been organized as follows. Section 2 describes the extraction of simple sentences from the parallel corpus given by TDIL. Section 3 and Section 4, describes the methodology for the training of the SMT and the NMT models respectively. Later, Section 5 and Section 6 describes the evaluation and conclusion, respectively.

2 Extraction of Simple Sentence Pairs

Since we wanted to analyze and compare both the models, SMT and NMT, with respect to how they perform on simple sentences, we first needed to extract such instances from our dataset that had data of varying complexity.

A simple sentence in this context is defined as a sentence which contains only one independent clause and has no dependent clauses. Generally, whenever two or more clauses are joined by conjunctions (coordinating and subordinating), it becomes a complex or a compound sentence accordingly. So, to get a hold on handling the conjunctions, we used the Stanford Dependency Parser [4] library to chunk the English sentences into phrases. (viz. NP (Noun Phrase), VP (Verb Phrase), PP (Preposition Phrase), ADJP (Adjective Phrase) and ADVP (Adverb Phrase)).

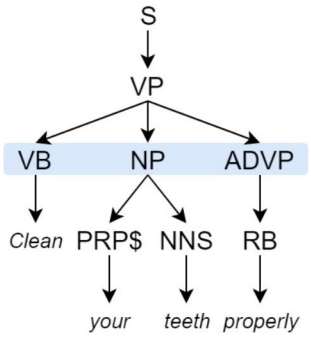


Figure 1: Extraction of phrase chunks.

We noticed that, simple sentences have an unique phrase structure that consists of combinations of NP, VP and PP. In conjunction with this theory, we applied two methods (viz. rule based approach and deep learning based approach) to extract simple sentences from the English corpus. The approaches are discussed in Section 2.1 and Section 2.2, respectively.

2.1 Rule Based Approach

We subjected 3046 simple sentences, extracted from various websites, to chunking using Stanford Dependency Parser (Manning et al., 2014), and extracted the unique phrase structures, which became the rules by which we further mined for simple sentences from the English corpus.

We extracted 205 unique rules, the surface forms of which, along with its Confidence Score, are shown in Table 1. The confidence score of the rules were calculated using

$$ConfidenceScore = \frac{No.OfSentencesPertainingToARule}{TotalNo.OfSentencesInTheTestData}$$

We tested our system on 2876 sentences (1438 simple sentences and 1438 complex/compound sentences) and got an accuracy of 89.22%. Table

[1] http://www.tdil.meity.gov.in/
[2] https://en.wikipedia.org/wiki/Devanagari
[3] https://en.wikipedia.org/wiki/Eastern_Nagari_script
[4] https://stanfordnlp.github.io/CoreNLP/

2 shows the various validation metrics. Using this system, 10,349 simple sentences from the TDIL English corpus was extracted, as shown in Table 4.

Rules	Confidence
PP NP* PP VP NP*	8.40
PP NP* VP PP NP*	9.49
ADVP NP* VP* ADVP NP*	9.36
NP VP PP NP PP NP	12.15
NP ADVP VP* NP*	11.69
NP* VP NP*	11.69
NP* PP NP VP* NP	11.46
NP VP PP NP*	11.23
VP* NP* PRP* ADVP*	4.92
NP VP* NP* PP* ADJP* ADVP*	9.62

Table 1: Surface forms of the extracted rules. "*" means one or more occurrence of item.

	Other	Simple	Prec.	Kappa
Other	1275	90	93.41%	0.78
Simple	220	1291		
Recall	85.28%			
Acc.	89.22%			
F1	89.16%			

Table 2: Confusion matrix for the rule based approach.

2.2 Deep Learning Based Approach

We preferred Deep Learning approach over traditional Machine Learning (ML) approach as because in the ML approach we could only extract syntactic features, which was already exploited in the rule based approach discussed in Section 2.1. On the other hand, a deep learning technique learn categories incrementally through its hidden layer architecture. We wanted the deep learning framework to learn from the POS tags itself as it automatically clusters similar data into separate spaces.

For the deep learning model, we trained a multilayer feed-forward neural network with stochastic gradient descent (Bottou, 2010) as optimizer with back-propagation. The network contained two hidden layers of sizes 50 and 50 respectively. The activation function used was *tanh* and loss function used was *Mean Squared Error*. *Learning Rate* was kept at 0.001 and *number of epochs* were fixed at 100. The *batch size* was kept at 128. The training data consisted phrases of 2876 sentences (1438 simple sentences and 1438 other complex/compound sentences). The trained model was subjected to 10 fold cross validation and it yielded an accuracy figure of 92.11%. Table 3 shows the other important validation metric measures.

	Other	Simple	Prec.	Kappa
Other	1287	76	92.22%	0.84
Simple	151	1362		
Recall	92.11%			
Acc.	92.11%			
F1	92.16%			

Table 3: Confusion matrix for deep learning based approach.

The TDIL English corpus was fed to this model and it yielded 14,976 simple sentences as shown in Table 4.

# of sentences		49999
# of other sentences	RL	39650
# of simple sentences	RL	10349
# of other sentences	DL	35023
# of simple sentences	DL	14976

Table 4: Simple Sentence Count

The deep learning based approach was preferred as it resulted in better accuracy. The Bengali and Hindi counterparts of these sentences were extracted to build a parallel corpus comprising of simple sentences only. The next step was to build MT models using this data, as well as the data from the whole corpus, and compare their respective results.

3 Statistical Machine Translation

Moses (Koehn et al., 2007) is a statistical machine translation system that allows us to automatically train translation models for any language pair, making use of a large collection of translated texts (parallel corpus). Once the model has been trained, an efficient beam search algorithm quickly finds the highest probability translation among the exponential number of choices.

For training the SMT model, we used English as the source language and Bengali and Hindi as the target languages. To prepare the data for training the SMT system, we performed the following steps.

3.1 Preprocessing

The following steps were employed to preprocess the Source and the Target texts.

- **Tokenization**: Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called tokens. In our case, these tokens were words, punctuation marks, numbers.
- **Truecasing**: This refers to the process of restoring case information to badly-cased or non-cased text (Lita et al., 2003). Truecasing helps in reducing data sparsity.
- **Cleaning**: Long sentences (# of tokens > 80) were removed.

3.2 Language Model

Post these steps a *Language Model* (LM) was built using the target language, Bengali and Hindi, in our case, to ensure fluent output. KenLM (Heafield, 2011), which comes bundled with the Moses toolkit, was used for building this model.

3.3 Word Alignment and Phrase Table Generation

For word alignment in the translation model, GIZA++ (Och and Ney, 2003) was used. Finally, the phrase table was created and probability scores were calculated. Training the Moses statistical MT system resulted in the generation of two models, one is a Phrase Model and the other is a Translation Model. Moses scores the phrase in the phrase table with respect to a given source sentence and produces best scored phrases as output.

The results of this system when trained and tested on the simple sentence corpus and the general corpus, for both En-Bn and En-Hn language pairs. The results and evaluation of the systems are shown in Sec 5, Table 5 and Table 6.

4 Neural Machine Translation

Neural machine translation (NMT) is an approach to machine translation that uses neural networks to predict the likelihood of a sequence of words, typically modeling entire sentences in a single integrated model. NMT departs from traditional phrase-based statistical approaches in that uses separately engineered subcomponents like Language Model generation, Word Alignment and Phrase Table generation. The main functionality of NMT is based on the sequence to sequence (seq2seq) architecture, which is described in Section 4.1.

4.1 Seq2Seq Model

The sequence to sequence model is a relatively new idea for sequence learning using neural networks. It has gained quite some popularity since it achieved state of the art results in machine translation task. Essentially, the model takes as input a sequence

$$X = \{x_1, x_2, ..., x_n\}$$

and tries to generate the target sequence as output

$$Y = \{y_1, y_2, ..., y_m\}$$

where x_i and y_i are the input and target symbols respectively. The architecture of seq2seq model comprises of two parts, the encoder and decoder. We experimented with two types of NMT models (word and character level), with both using the seq2seq architecture, the difference being in the inputs to its encoder and decoder. They are discussed in the sections 3 and 4 below. The working of seq2seq architecture at the word level is shown in Fig. 2. We implemented both the models using the Keras (Chollet et al., 2015) library.

4.1.1 Word Level NMT

To build our world level NMT model, we used the seq2seq with attention mechanism. This architecture has recently shown to achieve state of the art quality translation across many different language pairs. The details of the seq2seq model along with the training details are given below.

Encoder The encoder takes a variable length sequence as input and encodes it into a fixed length vector, which is supposed to summarize it's meaning and taking into account it's context as well. A Long Short Term Memory (LSTM) cell was used to achieve this. The directional encoder reads the sequence from one end to the other (left to right in our case),

$$\vec{h}_t = \vec{f}_{enc}(E_x(x_t), \vec{h}_{t-1})$$

Here, E_x is the input embedding lookup table (dictionary), $\vec{f}_{enc}$ are the transfer function for the Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) recurrent unit. A contiguous sequence of encodings C is constructed which is then passed on to the decoder.

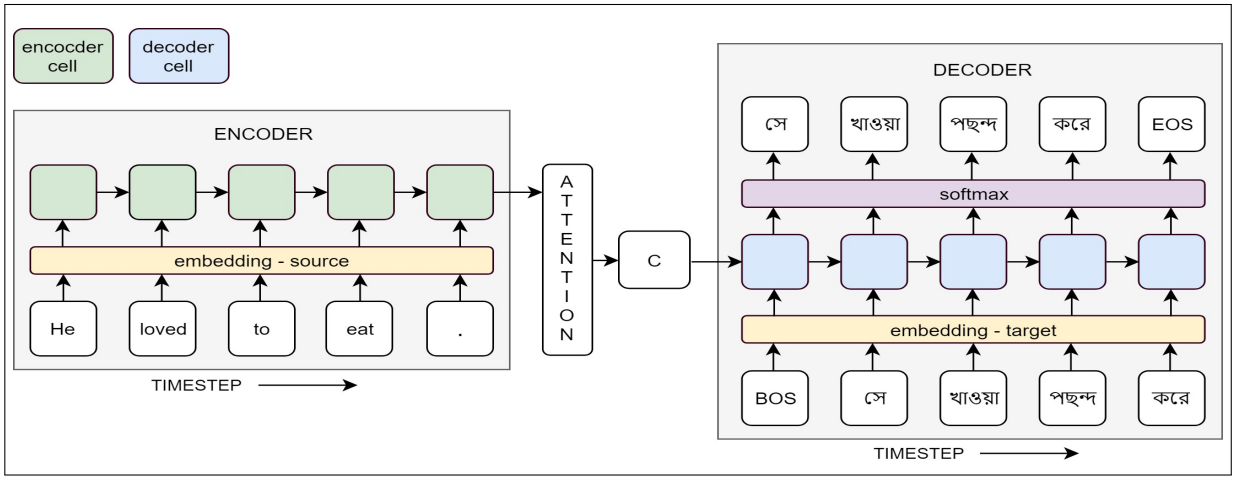


Figure 2: NMT with attention architecture.

Decoder The decoder takes as input, the context vector C from the encoder, and computes the hidden state at time t as,

$$s_t = f_{dec}(E_y(y_{t-1}), s_{t-1}, c_t)$$

Subsequently, a parametric function out_k returns the conditional probability using the next target symbol k.

$$(y_t = k \mid y < t, X) = \frac{1}{Z} exp(out_k(E_y(y_t - 1), s_t, c_t))$$

Z is the normalizing constant,

$$\sum_j exp(out_j(E_y(y_t - 1), s_t, c_t))$$

The entire model can be trained end-to-end by minimizing the log likelihood which is defined as

$$L = -\frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_y^n} logp(y_t = y_t^n, y_i t^n, X^n)$$

where N is the number of sentence pairs, and X^n and y_t^n are the input sentence and the t-th target symbol in the n-th pair respectively.

Training For training our model, we used the seq2seq with attention architecture with LSTM cell. We used two LSTM cells, stacked upon each other, where one acts as the encoder and the other as the decoder. We trained our model on 14976 data (for simple sentence corpus), 49999 sentences (for Bengali and Hindi general corpus), *batch size* at 256, *number of epochs* at 100 and *learning rate* at 0.001. The activation function used was *softmax*, optimizer used was *rmsprop* and the loss calculation at each step was done using *categorical cross-entropy*.

Attention Neural processes involving attention (Vaswani et al., 2017) has been largely studied in computational neuro-science. This concept is very loosely based on visual attention mechanism in humans. With attention mechanism, the need to encode the full source sentence into a fixed length vector is omitted. Rather we allow the decoder to attend different parts of the source sentence at each time step of the output generation. Essentially, we let the model learn what to *attent* to based on the input sequence and what is predicted so far.

Mathematically, it computes the context vector c_t at each time step t as a weighted sum of the source hidden states,

$$c_t = \sum_{t=1}^{Tx} \alpha_t h_t$$

Each attention weight α_t represents how much relevant the t-th source token x_t is to the t-th target token y_t and is computed as :

$$\alpha_t = \frac{1}{Z} exp(score(E_y(y_t - 1), s_{t-1}, h_t))$$

where

$$Z = \sum_{k=1}^{Tx} exp(score(E_y(y_t - 1), s_{t-1}, h_k))$$

Z is the normalization constant. score() is a feed forward neural network with a single hidden layer that scores how well the source symbol x_x and the target symbol y_t match. E_y is the target embedding lookup table and s_t is the target hidden state at time t.

The results and evaluation of the systems are shown in Sec 5.

4.1.2 Character Level NMT

Character level NMT (CNMT) performs better than Word Level NMT due to the following reasons

1. It does not suffer from out-of-vocabulary issues
2. It is able to model different, rare morphological variants of a word
3. It does not require segmentation (Chung et al., 2016).

Generally, CNMT works best when majority of alphabets, in the source and target language, overlap i.e both the languages share a common or similar script. Still, we tried to find out its performance on the simple sentence and whole corpus, though in our case Nagari script and Roman script utilizes completely different alphabets. The model has two parts (encoder and decoder) as discussed below.

Encoder For building the encoder we used LSTM cells. The input of the cell was one hot tensor of English sentences (embeddings at character level). From the encoder, the internal states of each cell were preserved and the outputs were discarded. The purpose of this is to preserve the information at context level. These states were then passed on to the decoder cell as initial states.

Decoder For building the decoder, again an LSTM cell was used with initial states as the hidden states from encoder. It was designed to return both sequences and states. The input to the decoder was one hot tensor (embeddings at character level) of Bengali and Hindi sentences while the target data was identical, but with an offset of one time-step ahead. The information for generation is gathered from the initial states passed on by the encoder. Thus, the decoder learns to generate target data [t+1,...] given targets [..., t] conditioned on the input sequence. It essentially predicts the output sequence, one character per output time step.

Training For training the model, *batch size* was set to 64, *number of epochs* was set to 100, activation function was *softmax*, optimizer chosen was *rmsprop* and loss function used was *categorical cross-entropy*. Learning rate was set to 0.001.

The results and evaluation of the systems are shown in Sec 5.

5 Evaluation and Analysis

All of our translation systems were evaluated in two ways, automatic and manual, depiction s of which are discussed in the section below.

5.1 Automatic Evaluation

Automatic evaluation was done by scoring the translations using BLEU and TER metrics. The results are shown in Table 5 and 6. In the tables, "Bn" and "Hn" means Bengali and Hindi respectively. "CNMT" and "WNMT" means character and word level NMT models respectively. Presence of Attention mechanism in the model is denoted using "A" and the contrary is denoted using "NA"

Model (Bn)	Simple Sent.		Whole Corp.	
	BLEU	TER	BLEU	TER
SMT	0	117.67	15.9	85.26
CNMT (NA)	8.69	91.87	4.19	88.22
WNMT (NA)	9.68	86.84	3.61	98.03
WNMT (A)	9.95	85.66	3.77	96.72

Table 5: Automatic evaluation metrics for En-Bn Model.

Model (Hn)	Simple Sent.		Whole Corp.	
	BLEU	TER	BLEU	TER
SMT	3.98	101.945	12.86	95.092
CNMT (NA)	7.98	92.85	5.96	85.18
WNMT (NA)	10.01	90.28	4.87	96.97
WNMT (A)	10.54	90.26	5.21	94.20

Table 6: Automatic evaluation metrics for En-Hn Model.

5.2 Manual Evaluation

Translation quality was judged by four linguists. Two had Bengali mother tongue (evaluated Bn model), while the other two had Hindi mother tongue (evaluated Hn model). The evaluation criterion were Adequacy and Fluency. Adequacy means how much of the meaning expressed in the target translation. Fluency means to what extent the translation is well-formed grammatically, contains correct spellings and intuitively acceptable and can be sensibly interpreted by a native speaker. The speakers were asked to rate the translation in range of 1-5, where '1' is the lowest and '5' is the highest. The manual evaluation measures for English-Bengali and English-Hindi language pair

Model (Bn)	SMT		CNMT		WNMT (NA)		WNMT(A)	
Corpus	**Simple**	**Whole**	**Simple**	**Whole**	**Simple**	**Whole**	**Simple**	**Whole**
Adequecy 1	0	2.15	1.98	1.54	2.02	1.44	2.15	1.47
Fluency 1	0	1.87	2.27	1.98	2.36	1.86	1.98	2.02
Adequecy 2	0	2.24	1.87	1.66	1.96	1.57	2.01	1.69
Fluency 2	0	1.92	2.05	1.86	2.21	1.77	2.26	1.93
Avg. Adequecy	0	2.195	1.925	1.6	1.99	1.505	2.08	1.58
Avg. Fluency	0	1.895	2.16	1.92	2.285	1.815	2.12	1.975

Table 7: Depiction of Manual Evaluation conducted by Bengali language speaking experts.

Model (Hn)	SMT		CNMT		WNMT(NA)		WNMT(A)	
Corpus	**Simple**	**Whole**	**Simple**	**Whole**	**Simple**	**Whole**	**Simple**	**Whole**
Adequecy 1	0.8	2.06	1.96	1.69	2.36	1.47	2.26	1.49
Fluency 1	0.5	1.72	2.04	2.08	2.27	1.92	2	2.22
Adequecy 2	1.02	2.18	1.79	1.71	2.02	1.63	2.18	1.9
Fluency 2	0.65	1.98	2.1	1.94	2.39	1.83	2.33	1.87
Avg. Adequecy	0.91	2.12	1.875	1.7	2.19	1.55	2.22	1.695
Avg. Fluency	0.575	1.85	2.07	2.01	2.33	1.875	2.165	2.045

Table 8: Depiction of Manual Evaluation conducted by Hindi language speaking experts.

is given in Table 7 and Table 8 respectively. In the tables, presence of Attention mechanism in the model is denoted using "A" and the contrary is denoted using "NA".

5.3 Analysis

We can clearly see in the results, that a NMT model, when trained using simple sentences, performs better than a SMT model, when trained using the same sentence pairs.

But, at the same time, SMT outperforms NMT, when trained using the whole corpus. This is due to the fact that NMT doesn't quite work well with less amount of data and highly complex sentences.

Similarly, we also see that character based NMT works better than word based NMT, when dealing with less amount of data. But again, we have to keep in mind that for a character based NMT to work well, we have to train it using a Source-Target language pair, who share a common script.

Further, word based NMT with attention perform relatively better than a character based NMT. We didn't use attention in the Character NMT, as attention won't be able to attent to individual characters.

6 Conclusion and Future Work

In this work, we have tried to analyze the scenarios where SMT performs better than NMT and vice-versa. Also, we have tried to find out whether MT models give better outputs when trained with simple sentences rather than when trained using sentences of various complexities.

As a future prospect, we would like to take the "other" (Complex+Compound) sentence pairs and simplify it, so that the whole MT models can be trained using more simple sentences. Also, we would like to increase the number of LSTM encoding and decoding layers as well as include embeddings like ConceptNet[5] in our future works.

Acknowledgments

This work is supported by Media Lab Asia, MeitY, Government of India, under the Visvesvaraya PhD Scheme for Electronics & IT.

References

Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, Dan Melamed, Franz-Josef Och, David Purdy, Noah A Smith, and David Yarowsky. 1999. Statistical machine translation. In *Final Report, JHU Summer Workshop*, volume 30.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

[5]https://github.com/commonsense/conceptnet-numberbatch

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

François Chollet et al. 2015. Keras. https://keras.io.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *CoRR*, abs/1603.06147.

Stephen Doherty, Sharon O?Brien, and Michael Carl. 2010. Eye tracking as an mt evaluation technique. *Machine translation*, 24(1):1–13.

Wei He, Zhongjun He, Hua Wu, and Haifeng Wang. 2016. Improved neural machine translation with smt features. In *AAAI*, pages 151–157.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*, volume 3, page 413.

Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Lucian Vlad Lita, Abe Ittycheriah, Salim Roukos, and Nanda Kambhatla. 2003. Truecasing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 152–159. Association for Computational Linguistics.

Shujie Liu, Nan Yang, Mu Li, and Ming Zhou. 2014. A recursive recurrent neural network for statistical machine translation.

Adam Lopez. 2008. Statistical machine translation. *ACM Computing Surveys (CSUR)*, 40(3):8.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

Sainik Mahata, Dipankar Das, and Santanu Pal. 2016. Wmt2016: A hybrid approach to bilingual document alignment. In *WMT*, pages 724–727.

Sainik Kumar Mahata, Dipankar Das, and Sivaji Bandyopadhyay. 2017. Bucc2017: A hybrid approach for identifying parallel sentences in comparable corpora. *ACL 2017*, page 56.

Sainik Kumar Mahata, Dipankar Das, and Sivaji Bandyopadhyay. 2018. Mtil2017: Machine translation using recurrent neural network on statistical machine translation. *Journal of Intelligent Systems*, pages 1–7.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. pages 311–318.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *EMNLP*, pages 1387–1392.

Warren Weaver. 1955. Translation. *Machine translation of languages*, 14:15–23.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Helping each Other: A Framework for Customer-to-Customer Suggestion Mining using a Semi-supervised Deep Neural Network

Hitesh Golchha, Deepak Gupta, Asif Ekbal, Pushpak Bhattacharyya
Indian Institute of Technology Patna
Patna, India
{hitesh.me14, deepak.pcs16, asif, pb}@iitp.ac.in

Abstract

Suggestion mining is increasingly becoming an important task along with sentiment analysis. In today's cyberspace world, people not only express their sentiments and dispositions towards some entities or services, but they also spend considerable time sharing their experiences and advice to fellow customers and the product/service providers with two-fold agenda: helping fellow customers who are likely to share a similar experience, and motivating the producer to bring specific changes in their offerings which would be more appreciated by the customers. In our current work, we propose a hybrid deep learning model to identify whether a review text contains any suggestion. The model employs semi-supervised learning to leverage the useful information from the large amount of unlabeled data. We evaluate the performance of our proposed model on a benchmark customer review dataset, comprising of the reviews of *Hotel* and *Electronics* domains. Our proposed approach shows the F-scores of 65.6% and 65.5% for the Hotel and Electronics review datasets, respectively. These performances are significantly better compared to the existing state-of-the-art system.

1 Introduction

The online platforms like social media websites, e-commerce sites of products and services, blogs, online forums and discussion forums etc. are very much attached today with our day-to-day lives.The availability of the these information sharing platforms has fueled the humans' desires to share one's opinions, emotions and sentiments with respect to the entities of all kinds: be it people, events, places, organizations, institutions, products, services, hobbies, games, movies, politics, technology etc. Generally people express their opinions in three ways: **(1)** through an independent piece of content writing **(2)** writing disposed towards a theme (such as a question in a community based question answering platform, or a topic in a discussion forum, or an entity in a product reviewing website/ e-commerce website) and **(3)** conversational writings in the form of exchange of utterances in dialog systems/chats or comments for a post in social media/online forums.

Such opinions which exist in different forms and places, have often hidden in them the experiences of people, their subjective emotions and sentiments towards different aspects of different entities, as well as the intentions of advices and suggestions proposing some action in a prescribed way. Suggestion mining can be thought of as a subproblem of opinion mining, entrusted with the task of extracting mentions of suggestions from the unstructured texts. Suggestions in the domain of reviews can be generally of two kinds:

1. **Customer to Companies:** These suggestions are directed from customers to the producers/service providers. Customers provide companies with feedbacks, often expressing their contentment or complaining about their dissatisfaction with certain product features, services, processes or amenities. They provide detailed reasons and personal experiences for the same and offer alternative ideas for implementation. These kinds of suggestions are not only important as a tool for the companies to review their current offerings, but they are also a great source of ideas for new directions.

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 196–205

2. **Customer to Customer:** These suggestions are provided from customers/users to the fellow customers/users. Customers share their experiences in reviews, and provide tips and recommendations to the other customers. This is sometimes more than merely the information whether they like some specific attributes of the products or services.

1.1 Motivation and Contributions

There are several use cases of automated retrieval and natural language understanding for suggestion mining. Apart from their own experiences, understanding and knowledge, people depend on the online community to form their own opinions and readily look for suggestions and tips from the other customers. The extracted suggestions and tips are equivalent to a set of effective guidelines for the other customers before they make their own decisions. The fellow users can avail more information, and hence the decision taken would be better. This is often beyond the sense conveyed by aspect based sentiment analysis (Thet et al., 2010; Gupta et al., 2015; Gupta and Ekbal, 2014).

Suggestions and feedbacks are also an important component of the market survey performed by the companies to drive innovation, change and improvements. This task is a prerequisite to other nuanced tasks which include classifying the domain of the suggestion, identifying the other arguments of the suggestions (finding the entity towards whom the suggestion is directed, identifying the aspects regarding which a suggestion has been made, finding the word boundaries of the suggestive expressions), and aggregation of such suggestions from multiple sources to comprehend a customer friendly summary.

We summarize the contributions of our proposed work as follows:

- We develop a linguistically motivated hybrid neural architecture to identify the review sentences that carry an intention of suggestion.
- We employ semi-supervised learning (self-training) along with a deep learning based supervised classification approach. This gives us the opportunity to harness the treasure of huge (unlabeled) data available in the form of customer reviews. To the best of our knowledge, this is the very first attempt in this direction to handle the target problem.
- Outperforming the current state-of-the-art customer-to- customer suggestion mining techniques and setting up a new state-of-the-art.

2 Related works

The field of suggestion classification and customer feedback analysis are relatively new in the area of Natural Language Processing (NLP) and Text Mining. Our work is most closely related to the prior research as reported in (Negi and Buitelaar, 2015; Negi et al., 2016). In (Negi and Buitelaar, 2015) authors defined the annotation guidelines for customer-to-customer suggestion mining. They trained a support vector machine (SVM) classifier over the features relevant for classification in the domains of hotels and electronics reviews. They used heuristic features, features extracted from the Part-of-Speech (PoS) tags, sequential pattern mining features, sentiment features and the features extracted from the dependency relations.
In their subsequent work, (Negi et al., 2016) demonstrated the improved performance using Convolutional neural networks (CNN) (Kim, 2014) and Long short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) based deep learning architectures to solve this problem. They experimented with both in-domain and cross-domain training data, and also compared their performance with a SVM based classifier trained with the same set of features similar to (Negi and Buitelaar, 2015).

There are some other existing works for suggestion mining, beyond customer-to-customer suggestions. (Ngo et al., 2017) developed a binary classification model based on Maximum Entropy and CNN for filtering suggestion intents in Vietnamese conversational texts like posts, comments, reviews, messages chat and spoken texts.

Brun and Hagege (Brun and Hagege, 2013) developed a feature-based suggestion mining system for the domain of product reviews. (Dong et al., 2013) performed suggestion mining on tweets of the customers regarding Microsoft Windows' phone. A model is proposed in (Wicaksono and Myaeng, 2013) which focused on extracting advices for the domains of travel using Hidden Markov Model (HMM) and Conditional Random Field (CRF). The work as reported in (Gupta et al., 2017) focused on classifying the customer feedback sentences of users into six classes using deep

learning based models.

Our proposed model differs from these existing works with respect to the problem addressed and the model developed. We have presented a very detailed comparison (in the experiments section) to the state-of-the-art system as reported in (Negi and Buitelaar, 2015; Negi et al., 2016).

3 Methodology

In this section at first we discuss the various deep learning models and then semi-supervised model.

3.1 Problem Definition

Given a multi-sentence review R having N sentences $\{s_1, s_2, \ldots, s_N\}$ the task is to categorize each sentence s_i into one of the classes $c \in C$, where C ={*"suggestive"*, *"non-suggestive* }. For a sentence s with a sequence $w_1, w_2, \ldots, w_{n-1}, w_n$ of n words, the associated suggestion class c can be computed as:

$$c = argmax_y p(y|s) \quad (1)$$

where $y \in C$

The following example review sentence, *"Tip if you want a beach chair at the beach or pool, go there before 9 am or so and put your magazine or towel on your chair."* is a "suggestion" intent directed towards a fellow customer. Here the expression of the intent is explicitly conveyed in the form of a review sentence with imperative mood[1]. The "non-suggestive" sentences instead contain statements and facts (e.g. **(1)** *"We stayed in the Westin Grand Berlin in July 2007."*) or expressions of one's sentiments (e.g. **(2)** *"But the rooms are small and not very functional.")*. An interesting thing to note is that the second example has implicit suggestions for the fellow customers as well as the service provider (hotel owner). The other visiting customers are implicitly advised against renting the rooms of the hotel as they are small and have less utility. Moreover, this review sentence also consists of an implicit suggestion to the hotel owner to offer larger rooms to their customers, and also improve the functionalities that they provide. However in our work, we only deal with the suggestions which are very explicitly mentioned, and that too directed specifically to the fellow customers.

3.2 Proposed Deep Learning Model

The customer-to-customer suggestion mining task requires recognizing specific syntactic and semantic constructions represented in texts. It should be able to capture the constructions representing imperative moods, and identify the patterns or phrases which are highly correlated with suggestive sentences in a review. It should also have a way for deep semantic understanding of text in order to disambiguate suggestions from the sentences which appear like suggestions on the surface.

We propose a hybrid model consisting of two deep learning based encoders designed to integrate different views or representations of the review sentences, and a linguistically motivated feature set. The information from the encoders along with linguistic knowledge are effectively combined with the help of a multi-layer perception (MLP) network. This is done to achieve higher abstraction necessary for a complex task like identifying the suggestive review sentence. Specifically, we use two different encoders, namely Convolutional Neural Network (CNN) and attention based Recurrent Neural Network (RNN). The effectiveness of CNN and RNN based encoder has been proven in other NLP tasks (Gupta et al., 2018d; Maitra et al., 2018; Gupta et al., 2018a,c). The CNN encoder uses multiple fully-connected over the convolution layer while the RNN encoder uses a LSTM layer with the attention (Raffel and Ellis, 2015) followed by multiple fully-connected layers. An overview of the architecture for suggestion mining is shown in Figure 1.

3.2.1 Linguistic Features

We use the following set of linguistic features in our model. We use slightly modified subset of features from (Negi and Buitelaar, 2015) and similar to (Gupta et al., 2018b)

Suggestive keywords : The suggestive keywords are usually associated with the texts containing actual suggestions. We use the following small set of suggestive keywords:
advice, suggest, may, suggestion, ask, warn, recommend, do, advise, request, warning, tip, recommendation, not, should, can, would, will
A binary-valued feature is defined that checks

[1]Imperative mood is a category or form of a verb which expresses a request or a command. For example, *"Get ready"*

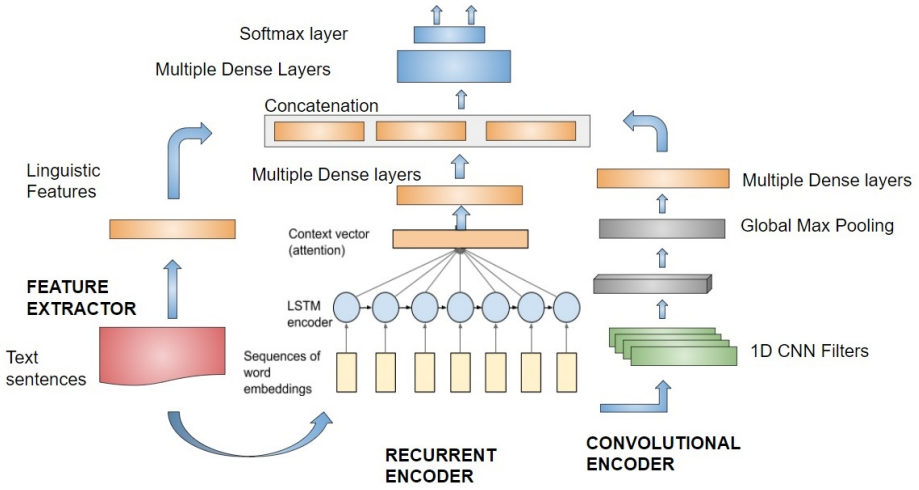


Figure 1: The proposed model architecture for customer suggestion mining

whether the current word is one of the keywords or not (1-presence, 0-absence).

N-gram features : We extract the most frequent 300 unigrams, 100 bigrams, and 100 trigrams from the training set. These are then used as a bag of n-gram features.

Part-of-Speech (PoS) N-gram features : We extract the most frequent PoS unigrams, bigrams and trigrams of size 50. These are then used as a bag of PoS n-grams features.

Imperative mood features : Most of the suggestions containing sentences have imperative mood. We try to capture this phenomenon by introducing the features obtained from the dependency trees[2]. We use the following imperative mood features:

1. Base verb (VB) at the beginning of sentence or without nsubj arc: In many imperative sentences, the subject (denoted by nsubj) is absent, i.e. it implies to be the second person. Moreover, the clause containing the suggestive expression begins with the base form of the verb (denoted by VB). Hence, this does not have any dependency relation with nsubj. This feature is illustrated in Figure 2.

2. 'nsubj' dependency relation features: The pair of PoS tags of the words connected by the dependency arc 'nsubj' is used as the bag of PoS feature. We describe the presence of this feature in Figure 3 and 4.

[2]We use spaCy dependency parser. For visualization, we used Stanford dependency parser

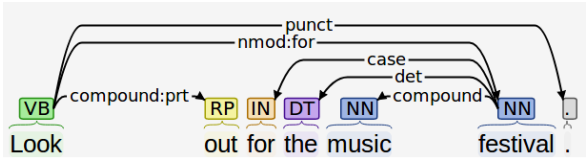


Figure 2: Presence of 'VB' without nsubj arc

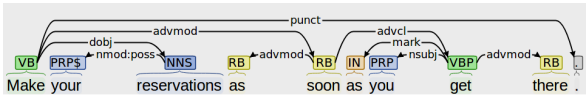


Figure 3: nsubj dependency arc relations. From this dependency tree the extracted features are (VBP, PRP).

This set of linguistic features are fed into a multilayer perceptron having two hidden layers of size 150 and 25, respectively.

3.2.2 Recurrent and CNN Encoders

The words in the sequence $\{w_1, w_2 \dots w_n\}$ from a given review sentence s are mapped to their corresponding word vectors $\{x_1, x_2 \dots x_n\}$. The word embeddings are obtained through the publicly available[3] GloVe word embeddings(Pennington et al., 2014) of dimension 300 and trained on the Common Crawl.

The recurrent encoder uses a LSTM network (hidden size 64) over the embedded sequences and it then applies an internal attention over the hidden states.

The LSTM network is able to process the sentence as a sequence, with the ability to capture long term dependencies. Thus the hidden layers

[3]https://nlp.stanford.edu/projects/glove/

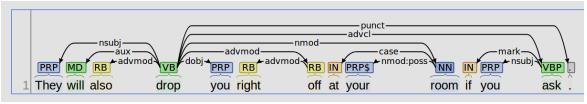


Figure 4: nsubj dependency arc relations. Here, (VB, PRP) and (VBP, PRP) features are active.

can efficiently perform composition over the local context, and help to identify patterns which are found in suggestive sentences. The attention mechanism then finds salient contexts and aggregates the important ones to build the context vector. The motivation for using attention stems from the fact that suggestive expressions can be identified in a short span of text within the sentence and the attention can effectively attend to those specific contexts encoded by LSTM. The Attention layer is followed by dense layers with 150, and 25 neurons, each having ReLU activations and a dropout value of 0.2.

The convolutional layer applies 250 one dimensional CNN filters of size 5 over the embeddings. The global max pooling is applied separately for the feature map obtained from each filter, and it helps to identify the presence of the n-gram feature corresponding to that feature in the sentence. The following dense layer with 250 neurons (ReLU activation and 0.75 dropout) helps to non-linearly compose multiple such features, thus giving itself an opportunity to learn a more diverse set of features.

3.2.3 Hybrid Model

The extracted linguistic features, the recurrent encoder representation and the convolutional encoder representation are concatenated (into a feature set p) and fed to a fully-connected layer with two neurons, followed by softmax activation. The softmax layer outputs the probability of the given review sentence being *suggestive* or *non-suggestive*. The probability that the output class $\hat{y}$ is i given the sentence s and parameters θ is computed as:

$$P(\hat{y} = i|s, \theta) = softmax_i(p^{\mathrm{T}} w_i + z_i) = \frac{e^{p^T w_i + z_i}}{\sum_{k=1}^{K} e^{p^T w_k + z_k}} \quad (2)$$

where z_k and w_k are the bias and weight vector of the k^{th} labels, p is the concatenated feature set, and K is the number of total classes (i.e. 2). θ is the set of all the parameters of the model. The system predicts the most probable class.

3.3 Semi-supervised Model

Semi-supervised learning makes use of both labeled (small) and unlabeled (huge) data for designing a more efficient classifier, as compared to the traditional supervised learning. We utilize self-training algorithm (Zhu, 2006), also known as bootstrapping, which can be flexibly used as a wrapper over any supervised learning algorithm. We use our hybrid model for this semi-supervised learning.

In self-training, we iteratively train a classifier enhancing each time the original training dataset with newly labeled instances. At the end of each iteration, the classifier is made to predict on the unlabeled dataset and 100 most confidently predicted instances of each class is added to the training data, with the predicted labels as the true labels. For self-training, a methodology similar to early stopping is applied, with a maximum of six iterations. We stop the iteration when the F1-Score on the validation data [4] does not improve over the existing best model in consecutive three iterations, saving only the best performing model for testing. For example in Fig. 5, the training terminates after the 6th iteration, and the model trained in the 3rd iteration is chosen for the final evaluation. Effect of adding unlabeled data to training for the electronics domain is depicted in Figure 6.

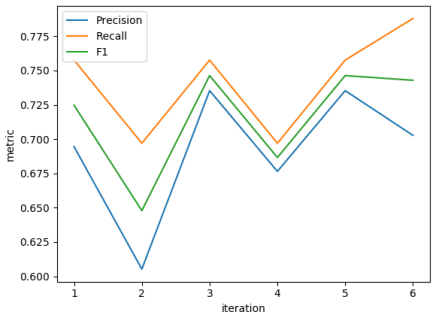


Figure 5: Scores on the validation set during self-training: Hotel domain.

For this semi-supervised setting, the cross-entropy error is minimized using the Adam Optimizer, and the training is stopped as the validation loss stops decreasing (early stopping). Because of the class imbalance (cf. Table 1), the loss function

[4] A part of the training set was used for validation.

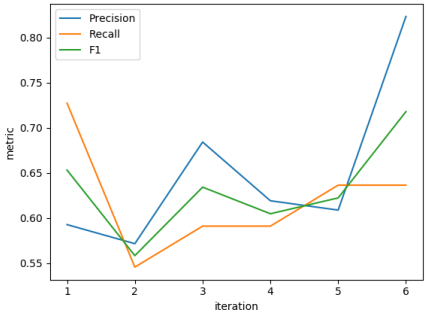


Figure 6: Scores on the validation set during self-training: Electronics domain.

weighs the loss for the positive class instances 10 times more than the loss for the negative class instances. All the other configurations are similar to the supervised setting[5].

4 Dataset and Experiments

In this section we present the datasets, experimental results and the necessary analysis.

4.1 Dataset

We conduct experiments on the dataset created by (Negi and Buitelaar, 2015)). The dataset comprises of the sentences of reviews taken from two domains, *viz.* Hotels and Electronics. The dataset was annotated as 'suggestive' and 'non-suggestive'.

The hotel reviews in (Negi and Buitelaar, 2015) are a subset of the TripAdvisor reviews annotated by (Wachsmuth et al., 2014)), with the sentiment polarity classes of positive, negative, neutral and conflicting. The electronics dataset was originally annotated by (Hu and Liu, 2004)) with the sentiment labels, and (Negi and Buitelaar, 2015) extended it for suggestion mining. The dataset consists of 7534 sentences from the hotel reviews and 3782 sentences from the reviews of electronic items. For semi-supervised learning experiments, we obtain the complete dataset from (Wachsmuth et al., 2014) for hotels. We segment these reviews into 21328 sentences in total. For the electronics domain, we use the Amazon reviews obtained from the electronics segment of (He and McAuley, 2016) as the unlabeled data. The first 50,000 sentences extracted from the reviews were chosen for the experiments.

[5]Models are optimized based on the validation set, a part of the training set

	Hotel reviews	Electronics reviews
Class = 1 *Suggestive*	407	273
Class = 0 *Non - Suggestive*	7127	3509
Total	7534	3782
Class1 : Class0	1:17.5	1:12.9

Table 1: Dataset statistics (on the sentence level)

Instances of suggestions and tips form a relatively small percentage of the total review sentences, and this is reflected in the class distribution of the labeled dataset. The number of instances is not enough for very deep architectures. Statistics of the datasets are presented in Table 1.

4.2 Results and Analysis

We re-implement the LSTM and CNN architectures proposed in (Negi et al., 2016) to construct our baselines. We re-implement this state-of-the-art system with the common training methodologies as ours. Detailed evaluation results are demonstrated in Table 2.

The LSTM is capable of handling long term dependencies and that may be attributed to its better performance against CNN for the domain of electronics where the average sentence length is relatively longer. The model based on LSTM achieves the F1 scores of 0.562 and 0.611 for the hotel and electronics datasets, respectively. CNN based model also demonstrates comparative performance with F1 scores of 0.598 and 0.600 for the two domains, respectively. Introducing attention to the LSTM model was found to be effective with reasonable performance improvement. Because of attention the system could attend to specific regions of the input sentence which had patterns similar to that of suggestive sentences, encoded by its query vector. The system with attention shows the best recall of 76.9% for the hotel reviews and 69.9% for the electronics reviews, establishing our claim about its ability. It achieves the F1 scores of 0.602 and 0.611 for the two domains, respectively.

Among the different architectures, the proposed hybrid model is found to be the best performing one with F1 scores of 0.643 and 0.621 for the two domains, respectively. Each of the encoders provides a different representation and features of the input, and the dense layers are able to combine them in an effective way. We also remove different encoders-one after other-from the proposed hybrid

Model	Hotel			Electronics		
	Precision	Recall	F1	Precision	Recall	F1
CNN	0.560	0.641	0.598	0.586	0.615	0.600
LSTM	0.511	0.624	0.562	0.582	0.644	0.611
LSTM + Attention	0.494	**0.769**	0.602	0.543	**0.699**	0.611
Negi and Buitelaar (2015)	0.580	0.512	0.567	**0.645**	0.621	0.640
Proposed Hybrid	0.593	0.703	0.643	0.587	0.660	0.621
Proposed Hybrid + Self Training	**0.639**	0.673	**0.656**	0.634	0.677	**0.655**

Table 2: Macro average evaluation results on 5-fold cross validation. Results of CNN and LSTM are based on the reimplementation of (Negi et al., 2016)

Model	Hotel			Electronics		
	Precision	Recall	F1	Precision	Recall	F1
Hybrid	0.593	0.703	0.643	0.587	0.660	0.621
Hybrid - CNN encoder	0.585	0.696	0.636	0.542	0.721	0.618
Hybrid - RNN encoder	0.636	0.641	0.638	0.586	0.644	0.614
Hybrid - Linguistic encoder	0.554	0.626	0.588	0.615	0.633	0.624

Table 3: Macro average evaluation on 5-fold cross validation for the ablation study of different component models

system to analyze the importance of each. Ablation studies of these models are reported in Table 3. For hotel reviews the order of importance of feature encoders are: *Linguistic encoder >CNN Encoder >RNN Encoder*. For electronics reviews the importance of model encoders are: *RNN Encoder >CNN Encoder >Linguistic Encoder*. Effectively, different representations of the review sentences and the corresponding features are indeed important for the classification task.

The use of self-training further improves the precision of the proposed hybrid model because it conservatively adds high confidence predictions obtained from the unlabeled data to the training data in each iteration. Inclusion of the 'suggestion' class examples into training helped in reducing the class imbalance, which leads to the improved recall scores for the positive class. Augmentation of new data also added more lexical variability for the system to learn. This, in turn, helps for better classification with the improved F1 scores of 0.656 and 0.655 for the hotel and electronics domains, respectively. The self training runs for a mean of 3.2 (SD = 0.75) iterations for the hotel domain, and 3.6 iterations (SD = 1.62) for the electronics domain. Thus, the expected number of unlabeled sentences added are 640 and 720, respectively. Our proposed system clearly performs better than the state-of-the-art model (Negi and Buitelaar, 2015) with the increments of 8.9 and 1.5 F1 score points for the hotel and electronics domain, respectively. Please note that the SVM based model was trained with a diverse and rich feature set. Statistical T-test show the performance improvement as significant.

5 Error Analysis

In order to understand the behaviors of our proposed model, we perform error analysis-both quantitatively and qualitatively. For quantitative analysis we depict the confusion matrix in Figure 7. Our closer analysis reveals that a lot of electron-

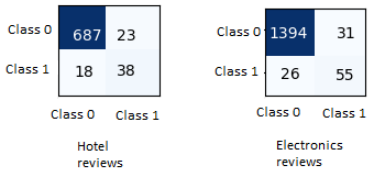


Figure 7: Confusion matrix on test set using Hybrid+Self- training. Here, 1: suggestive and 0: non-suggestive

ics reviews are slightly longer and more complex than the hotel reviews, making it slightly harder to predict despite having slightly more balanced class distribution. Moreover, the presence of only 273 reviews of class 1 in all (about 218 reviews in training), is too small for the architectures to effectively model.

We provide more detailed analysis with the actual examples. At first we describe the phenom-

ena where the instances are incorrectly predicted as suggestions (i.e. false positive cases):

- Many of the false positives are of imperative mood, but do not contain any suggestion towards any entity or product. For e.g.:
  Forget the fact that it will probably take me a year to figure out all the features this camera has to offer.
- Sometimes a user shares his/her own experience(s) but in the form of a second person, thus confusing the machine to predict as a suggestive sentence. For e.g.
  You book a top floor, you get first floor, you booked a suite, and got a room...you go out to your balcony to relax...and someone from a top floor....,which you reserved, has just spit on the back of your head.
- Many of the sentences consist of objects in the second person, but the sentences are not imperative in mood. Such errors are more common in the CNN based model, but are lesser in the proposed hybrid system that makes use of self-training. For e.g.
  If we find some great cheap places we will share it with you.
  "Sentence: very comfortable camera , easy to use , and the best digital photos you re going to get at this price"
- LSTM model sometimes incorrectly predicts those review texts where tokens with VB PoS tags appear. This happens because the sentence appears to be similar to a suggestive sentence that also starts from that particular word having VB PoS tag. For e.g.
  You need the storage to hold a decent amount of shots at 4 megapixel resolution
  might be confused with
  Hold a decent amount of ....
- Some of the false positives are actually suggestions which appear to be wrongly labeled in the original dataset. *We would definitely recommend this hotel to our friends.*
- Suggestions against a product/service which are sarcastic in nature have been annotated as non-suggestive but are difficult for our system to differentiate from the usual suggestions.
  I recommend this hotel only if you don't mind blithely throwing money around, and if you bring your own towels

We also show here few examples that contribute to the false negatives:

- When the sentences are very long, and only a clause of the text belongs to the imperative mood, it is missed by even the best system. For e.g. *"The battery lasts very long when playing music, but writing files to the player drains the battery fast , so you need to have it plugged into an outlet when sending files. "*
- Sometimes two sentences are clubbed together into one when the end marker is missing. In such scenarios, one of the sentences is suggestive and the other is not. In these cases the system predicts the sentence as non-suggestive. For e.g.
  "My only suggestion is to get a lens protector to help protect the shooting lens the lens coating will wear out after so many clean wipes and I m getting the those 52 mm adapter and uv lens filter at lensmateonline.com ."

  It becomes more tricky for the machine if the remaining part of the sentence contains multiple occurrences of first person pronoun. For e.g. *"You have to press the buttons hard and frequently I end up pressing enter when I meant to scroll ."*

From qualitative analysis we observe that systems have learned the ability to identify the sentences with suggestive terms and also the sentences which are imperative in nature. We believe that many of these errors can be reduced to a greater extent by increasing the size of the training data. With sufficient data, systems would be able to learn to better model the input, extract more relevant features, and be able to reason better about the differences between the suggestive sentences and sentences which look like suggestions.

6 Conclusion and Future Work

In this paper, we have proposed a hybrid deep learning model for the task of suggestion mining by incorporating richer and diverse representations of the inputs. We have also used self-training algorithm, which even improved the performance of the hybrid model, opening up more opportunities for the use of semi-supervised learning for this task. Experiments on benchmark datasets show that we obtain superior performance over the existing state-of-the-art system. In the future, we would like extend our work to other semi-supervised learning algorithms.

References

Caroline Brun and Caroline Hagege. 2013. Suggestion mining: Detecting suggestions for improvement in users' comments. *Research in Computing Science*, 70:199–209.

Li Dong, Furu Wei, Yajuan Duan, Xiaohua Liu, Ming Zhou, and Ke Xu. 2013. The automated acquisition of suggestions from tweets. In *AAAI*.

Deepak Gupta, Asif Ekbal, and Pushpak Bhattacharyya. 2018a. A Deep Neural Network based Approach for Entity Extraction in Code-Mixed Indian Social Media Text. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Deepak Gupta, Sarah Kohail, and Pushpak Bhattacharyya. 2018b. Combining graph-based dependency features with convolutional neural network for answer triggering. *arXiv preprint arXiv:1808.01650*.

Deepak Gupta, Pabitra Lenka, Harsimran Bedi, Asif Ekbal, and Pushpak Bhattacharyya. 2017. Iitp at ijcnlp-2017 task 4: Auto analysis of customer feedback using cnn and gru network. In *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 184–193. Asian Federation of Natural Language Processing.

Deepak Gupta, pabitra Lenka, Asif Ekbal, and Pushpak Bhattacharyya. 2018c. Uncovering code-mixed challenges: A framework for linguistically driven question generation and neural based question answering. In *Proceedings of 22nd International Conference on Computational Natural Language Learning . (CoNLL 2018)*. Association for Computational Linguistics (Accepted).

Deepak Gupta, Rajkumar Pujari, Asif Ekbal, Pushpak Bhattacharyya, Anutosh Maitra, Tom Jain, and Shubhashis Sengupta. 2018d. Can taxonomy help? improving semantic question matching using question taxonomy. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*, pages 499–513. Association for Computational Linguistics.

Deepak Kumar Gupta and Asif Ekbal. 2014. IITP: Supervised Machine Learning for Aspect based Sentiment Analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 319–323, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Deepak Kumar Gupta, Kandula Srikanth Reddy, Asif Ekbal, et al. 2015. PSO-ASent: Feature Selection using Particle Swarm Optimization for Aspect Based Sentiment analysis. In *International Conference on Applications of Natural Language to Information Systems (NLDB-2015)*, pages 220–233. Springer.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Anutosh Maitra, Shubhashis Sengupta, Deepak Gupta, Rajkumar Pujari, Asif Ekbal, Pushpak Bhattacharyya, Anutosh Maitra, Mukhopadhyay Abhisek, and Tom Jain. 2018. Semantic question matching in data constrained environment. In *Proceedings of the 21st International Conference on Text, Speech and Dialogue (TSD-2018)*, pages 499–513.

Sapna Negi, Kartik Asooja, Shubham Mehrotra, and Paul Buitelaar. 2016. A study of suggestions in opinionated texts and their automatic detection. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 170–178.

Sapna Negi and Paul Buitelaar. 2015. Towards the extraction of customer-to-customer suggestions from reviews. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2159–2167.

Thi-Lan Ngo, Khac Linh Pham, Hideaki Takeda, Son Bao Pham, and Xuan Hieu Phan. 2017. On the identification of suggestion intents from vietnamese conversational texts. In *Proceedings of the Eighth International Symposium on Information and Communication Technology*, pages 417–424. ACM.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Colin Raffel and Daniel PW Ellis. 2015. Feed-forward networks with attention can solve some long-term memory problems. *arXiv preprint arXiv:1512.08756*.

Tun Thura Thet, Jin-Cheon Na, and Christopher SG Khoo. 2010. Aspect-based sentiment analysis of movie reviews on discussion boards. *Journal of information science*, 36(6):823–848.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, Gregor Engels, and Tsvetomira Palakarska. 2014. A review corpus for argumentation analysis. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 115–127. Springer.

Alfan Farizki Wicaksono and Sung-Hyon Myaeng. 2013. Toward advice mining: Conditional random fields for extracting advice-revealing text units. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2039–2048. ACM.

X Zhu. 2006. Semi-supervised learning literature survey, department of computer sciences, university of wisconsin at madison, madison. Technical report, WI, Technical Report 1530. http://pages. cs. wisc. edu/˜ jerryzhu/pub/ssl_survey. pdf.

Towards Predicting Age of Acquisition of Words Using a Dictionary Network

Ditty Mathew*, **Girish Raguvir Jeyakumar***, **Rahul Kejriwal***, **Sutanu Chakraborti**

Department of Computer Science and Engineering
Indian Institute of Technology Madras
Chennai 600 036, India
ditty@cse.iitm.ac.in
{girishraguvir, kejriwalrahul.1996}@gmail.com
sutanuc@cse.iitm.ac.in

Abstract

Age of Acquisition (AoA) of words is an important psycho-linguistic property that influences various lexical tasks such as speed of reading words, naming pictures etc. In this paper, we study the effectiveness of graph theoretic and lexical features derived from dictionary networks in predicting the Age of Acquisition of English words. We show that dictionary networks contain a lot of information that can hint at the AoA of words, and the result promises that there are significant improvements over earlier approaches. We also extended the experiment to predict the Age of Acquisition of Hindi words and evaluated using words in Hindi textbooks.

1 Introduction

Age of Acquisition (AoA) (Kuperman et al., 2012) refers to the age at which a word is typically learned. For example, 'penguin' is generally learned earlier than 'albatross' and hence, 'penguin' has a lower age of acquisition value. Age of acquisition is an important feature for lexical decision tasks (Gilhooly and Logie, 1982) like speed of word reading, picture naming, word retrieval from lexical memory etc. AoA is important for two reasons. Firstly, word frequency is used as a feature alongside AoA in psycho-linguistic studies and is estimated from corpora consisting of materials meant for adult readers (Gerhand and Barry, 1999). Thus, word frequency typically underestimates the importance of words acquired earlier and thus, AoA plays an important role in complementing the information provided by word frequency. Secondly, it is believed that words acquired earlier have internal representations that can be activated faster independent of the number of times the word has been encountered. Despite the importance of the AoA parameter, the only way to estimate its value for any word is by surveying human participants. However, this process is in general tedious, requires active human participation and is thus limiting. We explore the possibility of an easier alternative. For this, we utilize the structure of word dictionary (Picard et al., 2013) where words are constructively defined.

Dictionaries provide recursive definitions and establish a dependency relation between words. To observe the inherent notion of order in which words are acquired, we consider a network constructed from a dictionary, which is a directed graph with words as vertices and edges denoting definitional participation. An edge from vertex A to vertex B implies that A is used in the definition of B in the dictionary. Thus, all predecessors of a vertex must be known for the vertex to be understood by a reader. The inherent idea is that some words (like "green") have to be learned via sensorimotor experience (Harnad, 1990), hence they are expected to be involved in loops/cycles in the network, while others can possibly be learned from constructive definitions composed of other known words which are probably simpler (Miller et al., 1990). Picard et al. (2013) extensively studied the hidden structure of the dictionary and used AoA to discriminate between different dictionary network regions successfully. This provides a strong motivation to explore the dictionary network structure for our problem - to estimate the AoA of words. In this paper, we build on this to take a step towards a faster alternative that can approximate AoA value for any word based on the dictionary network, and thus overcome the limited availability of data in addition to alleviating the need for tedious surveys.

* denotes equal contribution

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 206–214

It is important to keep in mind that the AoA is often highly subjective and can widely vary among subjects depending on their educational background, mental capabilities etc. So predicting AoA is in some way an ill-defined problem with an inherent noise in the data. Thus the primary motivation of the paper is to reveal the extent to which we can successfully predict AoA while restricting our scope to features derived from dictionary networks. It should be noted that aligning AoA may need richer cognitive and psycholinguistic features that are not contained in the dictionary network.

2 Background

2.1 Structure of Dictionary Network

Massé et al. (2008) developed a formal groundwork to determine "how word meaning is explicitly grounded in real dictionaries" and observed that meaning cannot be fetched based on dictionary definition recursively; at some point circularity of definitions must be broken by grounding the meaning of certain words. Massé et al. (2008) proposed a greedy algorithm to obtain a set of words in the dictionary network from which the rest of the words in the network can be defined without any circular dependencies. This set of words is called *grounding kernel*. The grounding kernel is estimated by repeatedly removing the nodes that do not have out-neighbors until there are no such nodes in the network. The intuition is that the nodes which do not have out-neighbors are not used for defining other words, so these words do not lead to circularity problem.

Picard et al. (2013) analyzed the full structure of the dictionary network, and they have found that the grounding kernel of dictionaries consist of a set of words, approximately 10% of the size of the entire dictionary, from which all other words can be defined. Inside the kernel, two sets of words are identified. One set is called *core* which includes the words in the strongly connected components (SCCs) that act as sources, i.e., all the words in these SCCs are defined by words in that SCC itself (in graph theoretical terms the nodes corresponding to such SCCs do not have incoming edges in its SCC-condensed graph[1]). The second set is called *satellites* which include rest of the SCCs in the grounding kernel.

[1]Each strongly connected component of a graph will be condensed to a single node in its SCC-condensed graph.

Vincent-Lamarre et al. (2016) studied the hierarchies of concepts in the concept network and proposed two types of hierarchies - kernel hierarchy and core hierarchy. These hierarchies are based on a graph theoretic property called *definitional distance*. In kernel hierarchy, the definitional distance of a word u is defined as,

1. $dist(u) = 0$, if $u \in kernel$
2. $dist(u) = 1 + \max\{dist(v) : v \in predecessors(u)\}$, otherwise

The words with definitional distance, $dist = 0$ are the words in the zeroth level of the kernel hierarchy. All words in the kernel of a dictionary will be at zeroth level. The words, which can be defined using the words in the zeroth level, are in the first level of the hierarchy and so on. For the words in the i^{th} level of the hierarchy, all the words which define these words should be at any level between 0 and $i-1$.

In core hierarchy, the definitional distance of words are computed with respect to core where core is a set of strongly connected components. The core hierarchy is a hierarchy of strongly connected components. The definitional distance of a word u from core is defined as,

1. $dist(u) = 0$, if $u \in core$
2. $dist(u) = 1 + \max\{dist(v) : \exists w \in SCC(u)$ such that $v \in predecessors(w)\}$, otherwise

where $SCC(u)$ contains all nodes in the strongly connected component that contains u.

2.2 Study of Psycho-linguistic Properties based on Dictionary Network

Vincent-Lamarre et al. (2016) studied the psycholinguistic properties such as frequency, concreteness, and age of acquisition of words with respect to the dictionary structure. This study is performed in dictionaries such as Cambridge dictionary, Longman dictionary, Merriam Webster dictionary and WordNet. Frequencies of words are computed using SUBTLEX (US) corpus; the concreteness ratings for 40,000 English word lemmas given in (Brysbaert et al., 2014) are used for concreteness; the age of acquisition ratings for 30,000 English words (Kuperman et al., 2012) are used for the age of acquisition property. They observed that the average age of acquisition of words that are part

of the core is smaller than the average age of acquisition of words that are part of satellites which is in turn smaller than the average age of acquisition of words that are not part of core and satellites. A similar trend is observed for frequency of words. For concreteness, the average concreteness value of words in satellites is observed to be smaller than the average concreteness value of words that are part of the core, which is in turn smaller than that of rest of the words. The properties such as frequency and age of acquisition follow the similar observation and it is hypothesized that words in the core are more frequent and acquired earlier compared to satellites, and words in the rest of the dictionary are less frequent and acquired later compared to satellites. This motivates us to test whether the dictionary structure will help in predicting the age of acquisition of words.

3 Dictionary Network Based Features

Let $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ be the directed graph constructed using words in the dictionary D. Each vertex $v \in \mathbb{V}$ indicates a word and each directed edge $(u, v) \in \mathbb{E}$ represents that the word u is used to define the word v in dictionary D. We propose graph-based features based on dictionary network $\mathbb{G}$ to predict the age of acquisition of each word defined in D.

We use the following basic dictionary structure-based features which were analyzed by Vincent-Lamarre et al. (2016) as having patterns related to AoA.

1. **Is core**: This is a binary value, which indicates whether the word is part of core or not. This feature is based on the work by Vincent-Lamarre et al. (2016) where they show that the words in the core are learned earlier than the satellite words, which are in turn learned earlier than the rest-of-dictionary.

2. **Is kernel**: This feature indicates whether the word is part of the kernel or not. This feature is motivated by the observation that the words in the kernel are acquired earlier than the words in the rest-of-dictionary (Vincent-Lamarre et al., 2016).

3. **Definitional distance from kernel and core**: These two features are defined by Vincent-Lamarre et al. (2016). For both definitional distances from core and kernel, Vincent-Lamarre et al. (2016) observed a linear trend between the definitional distance and the average age of acquisition of words at each definitional distance.

We propose the following features which are derived from the basic dictionary features studied by Vincent-Lamarre et al. (2016).

1. **PageRank**: The out-neighbors of a word w in $\mathbb{G}$ are the words which are defined using w. One would expect w to be acquired before words that are defined using w. Thus, *a word is acquired early if it is used to define several words that are acquired early; this leads to a circularity*. In order to resolve this circularity, we use the PageRank (Page et al., 1999) of words in the transpose graph $\mathbb{G}'$ whose vertices are same as $\mathbb{G}$, but edges are reversed to capture the importance from out-neighbors in $\mathbb{G}$.

2. **SCC PageRank**: The words within the strongly connected components (SCC) are closely associated. Because of circular dependencies between words in SCCs, it is hard to find which words are defined first. The correlation between definitional distance from core and AoA showed by (Vincent-Lamarre et al., 2016) claims that if there is an edge from the condensed node of SCC S_j to the condensed node of SCC S_i in the condensed graph, the words in S_i would be acquired earlier than words in S_j. In order to analyze the random walk interpretation of this property, we consider the importance of the SCC to which each word is associated as a feature and it is estimated as the PageRank of the corresponding condensed node in the condensed graph.

3. **Within SCC PageRank**: This feature computes the importance of a word within the SCC it belongs to as given by PageRank. We consider this feature to complement the SCC PageRank feature.

We propose the following local features to understand their trends in dictionary network for AoA prediction task.

1. **Word length**: The number of characters in the word is used as a feature. Generally, at lower ages, shorter words are learned and longer words tend to be acquired at later ages.

2. **In-degree centrality**: The in-degree centrality of a word is computed by dividing its in-degree by maximum possible in-degree. In-degree of a word w is a measure of the number of other words that are used in the definition of w. The intuition behind this is that a larger value suggests that this word may be acquired later.

3. **Out-degree centrality**: The out-degree centrality is the ratio of the out-degree of a word to the maximum possible out-degree. The out-degree of a word w is a measure of the number of words that are defined using w. The intuition is that the larger value should mean that this word may be acquired earlier.

4. **Local clustering coefficient**: The local clustering coefficient is the number of edges between immediate predecessors and successors divided by the maximum number of possible edges among them. This feature is used to study the effect of clustering tendencies on the AoA of a word.

5. **Second in-neighborhood and out-neighborhood**: The number of predecessors at distance 1 or 2 from a word in the graph is taken as the second in-neighborhood. Similarly, second out-neighborhood is the number of successors at distance 1 or 2 from a word in the graph. These features are used to study the higher level significance of in-degree centrality and out-degree centrality.

4 Experimental Evaluation

4.1 Datasets

We use the age of acquisition data from the Kuperman's (Kuperman et al., 2012) dataset and MRC psycho-linguistic dataset (Coltheart, 1981). The Kuperman dataset[2] contains AoA values for 31,124 words which are collected using Amazon Machine Turk. It contains data aggregated by asking participants to give one value in the range 1 to 25 for each word. The final AoA for a given word is then computed by taking the average of all the responses. The MRC psycho-linguistic dataset[3] contains lexical, morphological and psycho-linguistic properties of 1,50,837 words out of which the age of acquisition of 1,903 words are available. We use the age of acquisition ratings from both Kuperman and MRC datasets for evaluation.

We construct dictionary networks from Cambridge International Dictionary of English (CIDE), Longman Dictionary of Contemporary English (LDOCE), Merriam-Webster (MWC) dictionary and WordNet. We use the first definition of first sense in these dictionaries to build the corresponding dictionary graph. The stop words in the word definition are removed. The number of words in the network that is constructed from all four dictionaries are given in Table 1.

Dictionary	No of Words
CIDE	19,614
LDOCE	26,859
MWC	79,979
WordNet	76,792

Table 1: No of words in the network constructed from all four dictionaries

The graph-based features are extracted from these dictionary networks for the words that are common in all dictionaries and AoA dataset. There are 14,436 common words for Kuperman dataset, and we obtain 1,495 common words for MRC dataset.

4.2 Experiment Setup

We propose to use the dictionary network-based features along with richer cognitive and psycho-linguistic features for the prediction of age of acquisition of words. Hence, we use dictionary network-based features along with the lexical features and semantic features proposed in Paetzold and Specia (2016a) to predict the age of acquisition of words. The lexical features include

- Number of syllables
- Word's frequency in the Brown (Francis and Kucera, 1979), SUBTLEX (Brysbaert and New, 2009), SubIMDB (Paetzold and Specia, 2016b), Wikipedia, Simple Wikipedia (Kauchak, 2013) corpora
- Number of senses, synonyms, hypernyms, and hyponyms for word in WordNet

[2]Kuperman dataset is downloaded from http://crr.ugent.be/archives/806

[3]MRC psycho-linguistic dataset is downloaded from http://websites.psychology.uwa.edu.au/school/MRCDatabase/uwa_mrc.htm

- Minimum, maximum and average distance between the word's senses in WordNet and the root sense
- Number of images found for word in the Getty Image database[4]

The semantic features are the word embedded vectors (Mikolov et al., 2013) of words. The word embedded vectors capture the semantic information of words.

We train a ridge regression model to predict the AoA of words and train the model using lexical, semantic and dictionary features. We train the model using different sets of features,

1. Lexical features (Paetzold and Specia, 2016a)
2. Lexical and Semantic features (Paetzold and Specia, 2016a)
3. Lexical, Semantic and Dictionary network features based on CIDE dictionary
4. Lexical, Semantic and Dictionary network features based on LDOCE dictionary
5. Lexical, Semantic and Dictionary network features based on MWC dictionary
6. Lexical, Semantic and Dictionary network features based on WordNet

We use the lexical features and semantic features proposed in Paetzold and Specia (2016a) as a baseline to predict the age of acquisition of words. We compare the baseline model with the model that uses lexical features, semantic features, and dictionary features. We used the word embedded vectors trained using Google news dataset[5].

The model is evaluated by analyzing the Spearman's (ρ) (Spearman, 1906), Pearson's (r) (Pearson, 1920) and Kendall's tau (Sen, 1968) correlation coefficients between the actual AoA ranking and the predicted AoA ranking of words in test data. The Spearman's correlation is a measure of rank correlation and it assesses the monotonic relationships between variables. The Pearson's correlation measures the linear correlation between two variables. The Kendall's tau coefficient measures the association of variables based on pairwise ordering.

[4] http://developers.gettyimages.com/

[5] https://code.google.com/archive/p/word2vec/

4.3 Evaluation

We analyze the correlation coefficients on a 10-fold train-test splits and the average Spearman's (ρ), Pearson's (r), Kendall's tau correlations of test data are obtained. We compare the models which use dictionary network-based features with the baselines which do not use dictionary network-based features. The correlation coefficients of the predicted value with respect to the Kuperman dataset for all feature sets are listed in Table 2. All correlation coefficients are statistically significant with $p < 0.05$.

Features	Avg Spearman	Avg Pearson	Avg Kendall's tau
Lexical Features (Paetzold and Specia, 2016a)	0.4837	0.5049	0.3353
Lexical + Semantic Features (Paetzold and Specia, 2016a)	0.7793	0.7835	0.5856
Lexical + Semantic + CIDE	**0.7926***	**0.8004***	**0.5986***
Lexical + Semantic + LDOCE	0.7910*	0.7990*	0.5970*
Lexical + Semantic + MWC	0.7837*	0.7887*	0.5899*
Lexical + Semantic + WordNet	0.7878*	0.7924*	0.5941*

Table 2: Correlation Coefficients between the predicted AoA of words and the AoA based on Kuperman dataset; * indicates improvements are statistically significant with $p < 0.05$

We can observe that the model which uses dictionary network features result in better correlations compared to both baselines. The improvements in correlations are statistically significant with $p < 0.05$.

The correlation coefficients between the predicted AoA of words and the age of acquisition values in MRC psycho-linguistic dataset using all sets of features are given in Table 3. The correlation coefficients obtained when trained with all features are consistently performing better in all four dictionaries compared to baseline models.

Some words which are predicted as acquired earlier compared to its actual AoA using lexical and semantic (Paetzold and Specia, 2016a) features have their AoA predicted better when dictionary features are used. These words are observed as non-kernel words in the dictionary network. We also observed that the AoA prediction error is very less for words with a lower age of acquisition such as 'give', 'work', 'show', 'day' etc. when dictionary features are used compared to the baseline features. The inclusion of lexical and semantic features improved the prediction value for

Features	Avg Spearman	Avg Pearson	Avg Kendall's tau
Lexical Features (Paetzold and Specia, 2016a)	0.5632	0.5514	0.3993
Lexical + Semantic Features (Paetzold and Specia, 2016a)	0.7576	0.7700	0.5718
Lexical + Semantic + CIDE	0.7867*	**0.7957***	**0.6008***
Lexical + Semantic + LDOCE	**0.7880***	0.7956*	0.6005*
Lexical + Semantic + MWC	0.7609	0.7713	0.5746
Lexical + Semantic + WordNet	0.7703*	0.7820*	0.5830*

Table 3: Correlation Coefficients between the predicted AoA of words and the AoA based on MRC psycho-linguistic dataset; * indicates improvements are statistically significant with $p < 0.05$

Features	Avg Spearman	Avg Pearson	Avg Kendall's tau
Lexical Features (Paetzold and Specia, 2016a)	0.5673	0.4956	0.4004
Lexical + Semantic Features (Paetzold and Specia, 2016a)	0.7670	0.7358	0.5707
Lexical + Semantic + CIDE	0.7987*	**0.7672***	0.6026*
Lexical + Semantic + LDOCE	**0.8042***	0.7654*	**0.6068***
Lexical + Semantic + MWC	0.7741*	0.7467*	0.5783*
Lexical + Semantic + WordNet	0.7785*	0.7572*	0.5836*

Table 4: Correlation Coefficients between the predicted familiarity of words and the familiarity rating based on MRC psycho-linguistic dataset; * indicates improvements are statistically significant with $p < 0.05$

words which are i) general words like 'something', 'everyday', ii) colloquial words like 'grandma', 'mama', 'papa'. This is because of the infrequent use of such words in dictionary definitions.

We also study the prediction of other psycho-linguistic properties using dictionary network-based features. We experimented with the psycho-linguistic properties such as familiarity and imagability (Gilhooly and Logie, 1980). The familiarity of a word is the frequency with which a word is seen, heard and used. The imagability of a word is the intensity with which a word arouses images. We use the familiarity and imagability ratings presented in MRC psycho-linguistic dataset (Coltheart, 1981). This dataset contains familiarity ratings for 3,814 words and imagability ratings for 3,733 words. We trained a ridge regression model with the target value as the familiarity rating for predicting the familiarity of a word and the imagability rating is used as the target value for predicting the imagability of a word.

The same set of features are used for comparison and observed that both familiarity and imagability ratings are correlating better when dictionary network based features are used. All correlation coefficients are statistically significant with $p < 0.05$. The improvements in correlations are also statistically significant with $p < 0.05$. The dictionary features based on CIDE and LDOCE dictionaries are performing better than other two dictionaries.

Features	Avg Spearman	Avg Pearson	Avg Kendall's tau
Lexical Features (Paetzold and Specia, 2016a)	0.4734	0.4681	0.3252
Lexical +Semantic Features (Paetzold and Specia, 2016a)	0.7829	0.7708	0.5857
Lexical +Semantic + CIDE	**0.7915***	**0.7807***	**0.5939***
Lexical +Semantic + LDOCE	0.7912*	0.7797*	0.5936*
Lexical +Semantic + MWC	0.7860*	0.7757*	0.5877*
Lexical +Semantic + WordNet	0.7890*	0.7785*	0.5914*

Table 5: Correlation Coefficients between the predicted imagability of words and the imagability rating based on MRC psycho-linguistic dataset; * indicates improvements are statistically significant with $p < 0.05$

4.4 Experiment using School Textbooks

In this experiment, our task is to predict the class or grade[6] at which a word can be introduced in the school curriculum. We used the words present in Indian English textbooks from standard 1 to standard 10 which are published by National Council of Educational Research and Training (NCERT)[7]. If a word is first mentioned in i^{th} standard where $1 \leq i \leq 10$, then the target value of that word is i. In this way, we labeled the target value of words that are used in standard 1 to 10 English textbooks. We extracted all words from these textbooks and out of which we used the 5,496 words that are common in all four dictionaries.

For this task, we prefer regression over classification as the target values are ordered. We trained

[6] we use the words 'class', 'grade', or 'standard' interchangeably.

[7] http://www.ncert.nic.in/ncerts/textbook/textbook.htm

a ridge regression model to predict the target value and evaluated the model using mean squared error over 10-fold train-test splits. The model is trained using all sets of features and the results are given in Table 6. The mean squared error is highest when trained using only lexical features and it is decreased when lexical and semantic features are used. The mean squared error is even reduced when dictionary network-based features are used along with lexical and semantic features. The error is minimum when dictionary network features from LDOCE dictionary are used. The reduction in mean squared error is statistically significant with $p < 0.05$ when dictionary network-based features from CIDE, LDOCE and WordNet dictionaries are used and the deduction is statistically significant with $p < 0.01$ when MWC dictionary based features are used.

Features	Mean Squared Error
Lexical Features (Paetzold and Specia, 2016a)	0.8039
Lexical + Semantic Features (Paetzold and Specia, 2016a)	0.6621
Lexical + Semantic + CIDE	0.6326*
Lexical + Semantic + LDOCE	**0.6289***
Lexical + Semantic + MWC	0.6595+
Lexical + Semantic + WordNet	0.6553*

Table 6: Average Mean squared error when predicted the standard at which a word is first introduced to school students in NCERT English textbooks; * indicates deduction in mean squared error is statistically significant with $p < 0.05$ and + indicates deduction is statistically significant with $p < 0.01$

We also extended our experiment for Hindi words by using Hindi textbooks published by NCERT. Similar to the experiment using English Words, here the task is to predict the class or standard at which a Hindi word can be introduced in the school curriculum. We used the gloss of Hindi words in Hindi WordNet (Bhattacharyya, 2017) to construct the dictionary network using which the network based features are extracted. For lexical features, we used the frequency of words based on Hindi Corpus[8] published by Center for Indian Language Technology, IIT Bombay. The Hindi WordNet is used for other lexical features based on WordNet. We used Hindi Wikipedia[9] dump to train the embedded vectors of Hindi words. We used these features to train a ridge regression model to predict the standard at which a Hindi word can be introduced. We extracted words from NCERT Hindi textbooks from standard 1 to 10 and all features are obtained for 3,860 words. The model is evaluated using mean squared error over 10-fold train-test splits. The average mean squared error obtained when trained using i) only lexical features, ii) lexical and semantic features, iii) lexical, semantic and dictionary network features are given in Table 7. We can observe that the addition of dictionary network features improves the prediction compared to lexical and semantic features. This experiment also signifies the impact of the proposed model in predicting the age of acquisition of Hindi words for which an AoA dataset is not available.

Features	Mean Squared Error
Lexical Features (Paetzold and Specia, 2016a)	0.8708
Lexical + Semantic Features (Paetzold and Specia, 2016a)	0.8056
Lexical + Semantic + Dictionary	**0.7674***

Table 7: Average Mean squared error when predicted the standard at which a word is first introduced to school students in NCERT Hindi textbooks; * indicates deduction in mean squared error is statistically significant with $p < 0.05$

Class	Words
1	साथ (saath), काम (kaam), याद (yaad), फूल (phuul), जानवर (jaanvar)
2	आराम (aaram), दौरान (dauran), घास (ghas), असर (asar), पुलिस (pulice)
3	देखभाल (dekhbhaal), मुख (mukh), झलक (jhalak), वार (vaar)
4	दिमाग (dimaag), ट्रेन (train), जीव (jeev), उपहार (upahaar), खैर (khair)
5	पकवान (pakvaan), नौकर (naukar), बरतन (bartan), नमकीन (namkeen), मांग (maang)
6	जूट (juut), ढोलक (dholak), अधिगम (adhigam), आंगन (aangan), आश्वासन (aaswaasan)
7	गर्जन (garjan), खड्ग (khadag), जुल्म (julm), विकर्ण (vikarn), सूर्यदेव (sooryadev)
8	प्रेरक (prerak), आत्मसम्मान (aatmasamman), समाहित (samaahit), प्रतिमान (praratimaan)
9	अवमूल्यन (avamuulyan), वैष्णव (vaishnav), जागीरदार (jaageeradhaar), अध्यात्म (adhyaathm)
10	पाठ्यपुस्तक (paathyapustak), पूँजीवाद (punjeevaad), मूर्तिकार (muurtikaar), विद्याधर (vidhyaadhar)

Table 8: Examples of Hindi words predicted for Class/Standard 1 to 10

In order to qualitatively analyze the results, the predicted value of words is rounded when trained using all features. The class at which a word can be introduced is correctly predicted when the rounded predicted value is the same as the actual value assigned to the word. Some examples of Hindi words with the class/standard are correctly predicted are

[8] http://www.cfilt.iitb.ac.in/Downloads.html
[9] https://hi.wikipedia.org/

given in Table 8.

5 Discussion

Vajjala and Meurers (2014) proposed psycho-linguistic features such as the age of acquisition, familiarity, imagability and concreteness for predicting the reading level of a text and it is used for assessing the relative reading level of sentence pairs for text simplification. Our experiments suggest that the dictionary network-based features can be used as lexical features for predicting the reading level of a text.

In our experiments, the dictionary network-based features are performing better when extracted from Cambridge (CIDE) and Longman (LDOCE) dictionaries compared to Merriam Webster (MWC) and WordNet dictionaries. The motivation of proposing dictionary network-based features for predicting age of acquisition is the observation by Vincent-Lamarre et al. (2016) that the average age of acquisition of words in the core is smaller than the average age of acquisition of words in the satellites which is in turn smaller than the average age of acquisition of words in the rest of the dictionaries. In Figure 1, the average age

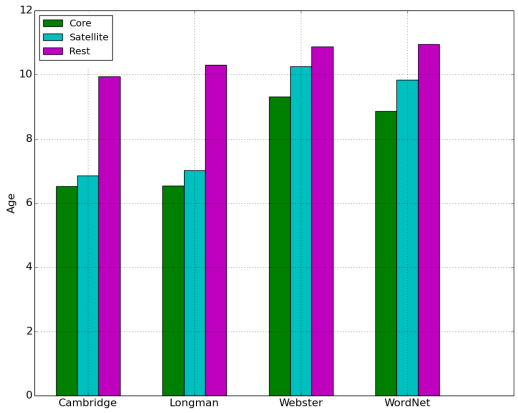


Figure 1: Average age of acquisition of words in Core, Satellites and Rest of the words in all four dictionaries

of acquisition values of words in core, satellites and the rest of the words are given for all four dictionaries. We can observe that the average AoA values are increasing from core to satellites to rest of the words in all four dictionaries. The average AoA of words in the kernel is the average of the AoA of words in core and satellites. From Figure 1, it is clear that the average AoA of kernel words will be smaller than the rest of the words. We can observe that the differences between the average AoA value of core words and the rest of the words are small in MWC and WordNet dictionaries, whereas the differences are large in CIDE and LDOCE dictionaries. This may be the reason that the CIDE and LDOCE dictionaries are performing better than MWC and WordNet dictionaries.

Concreteness (Brysbaert and New, 2009) is another psycho-linguistic property which is widely used along with age of acquisition for reading level prediction, text simplification etc. Concreteness is the extent to which the object that the word can be experienced by senses. The proposed dictionary network features are based on the dictionary structure where core words are defined before satellite words which are in turn defined before the rest of the words in the dictionary. Vincent-Lamarre et al. (2016) observed that the words present in the satellites are more concrete than the words present in the core. Since concreteness does not follow the progression from core to satellite and then to the rest of the words, which is central to our hypothesis, we did not use dictionary features to estimate concreteness.

6 Conclusion

We study the effectiveness of graph-theoretic features derived from dictionary networks in predicting the age of acquisition of words and the result shows significant improvements over earlier approaches that relate dictionary features to AoA. This work is a step towards understanding the difficult cognitive task of understanding how we acquire words. We also study the usefulness of dictionary network-based features for predicting other psycho-linguistic properties such as familiarity and imagability and the results are promising.

To the best of our knowledge, the psycho-linguistic study on Hindi has not been done before. Our experiment on Hindi words signifies the impact of the proposed model in predicting the age of acquisition of Hindi words.

References

Pushpak Bhattacharyya. 2017. IndoWordNet. In *The WordNet in Indian Languages*, pages 1–18. Springer.

Marc Brysbaert and Boris New. 2009. Moving beyond Kučera and Francis: A Critical Evaluation of Current Word Frequency Norms and the Introduction of a New and Improved Word Frequency Measure for American English. *Behavior Research Methods*, 41(4):977–990.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness Ratings for 40 Thousand Generally Known English Word Lemmas. *Behavior Research Methods*, 46(3):904–911.

Max Coltheart. 1981. The MRC Psycholinguistic Database. *The Quarterly Journal of Experimental Psychology*, 33(4):497–505.

WN Francis and H Kucera. 1979. Brown Corpus Manual. *Brown University*.

Simon Gerhand and Christopher Barry. 1999. Age of Acquisition, Word Frequency, and the Role of Phonology in the Lexical Decision Task. *Memory & Cognition*, 27(4):592–602.

Ken J Gilhooly and Robert H Logie. 1980. Age-of-acquisition, Imagery, Concreteness, Familiarity, and Ambiguity Measures for 1,944 Words. *Behavior Research Methods & Instrumentation*, 12(4):395–427.

KJ Gilhooly and RH Logie. 1982. Word Age-of-Acquisition and Lexical Decision Making. *Acta Psychologica*, 50(1):21–34.

Stevan Harnad. 1990. The Symbol Grounding Problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346.

David Kauchak. 2013. Improving Text Simplification Language Modeling using Unsimplified Text Data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1537–1546.

Victor Kuperman, Hans Stadthagen-Gonzalez, and Marc Brysbaert. 2012. Age-of-acquisition Ratings for 30,000 English Words. *Behavior Research Methods*, 44(4):978–990.

A Blondin Massé, Guillaume Chicoisne, Yassine Gargouri, Stevan Harnad, Olivier Picard, and Odile Marcotte. 2008. How is Meaning Grounded in Dictionary Definitions? In *Proceedings of the 3rd Textgraphs Workshop on Graph-Based Algorithms for Natural Language Processing*, pages 17–24. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to WordNet: An On-line Lexical Database. *International Journal of Lexicography*, 3(4):235–244.

Gustavo Paetzold and Lucia Specia. 2016a. Inferring Psycholinguistic Properties of Words. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 435–440.

Gustavo H Paetzold and Lucia Specia. 2016b. Unsupervised Lexical Simplification for Non-Native Speakers. In *AAAI*, pages 3761–3767.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford InfoLab.

Karl Pearson. 1920. Notes on the History of Correlation. *Biometrika*, 13(1):25–45.

Olivier Picard, Mélanie Lord, Alexandre Blondin-Massé, Odile Marcotte, Marcos Lopes, and Stevan Harnad. 2013. Hidden Structure and Function in the Lexicon. *arXiv preprint arXiv:1308.2428*.

Pranab Kumar Sen. 1968. Estimates of the Regression Coefficient based on Kendall's tau. *Journal of the American Statistical Association*, 63(324):1379–1389.

Charles Spearman. 1906. 'Footrule' for Measuring Correlation. *British Journal of Psychology*, 2(1):89–108.

Sowmya Vajjala and Detmar Meurers. 2014. Assessing the Relative Reading Level of Sentence Pairs for Text Simplification. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 288–297.

Philippe Vincent-Lamarre, Alexandre Blondin Massé, Marcos Lopes, Mélanie Lord, Odile Marcotte, and Stevan Harnad. 2016. The Latent Structure of Dictionaries. *Topics in Cognitive Science*, 8(3):625–659.

Towards Predicting Age of Acquisition of Words Using a Dictionary Network

Ditty Mathew*, **Girish Raguvir Jeyakumar***, **Rahul Kejriwal***, **Sutanu Chakraborti**

Department of Computer Science and Engineering
Indian Institute of Technology Madras
Chennai 600 036, India
ditty@cse.iitm.ac.in
{girishraguvir, kejriwalrahul.1996}@gmail.com
sutanuc@cse.iitm.ac.in

Abstract

Age of Acquisition (AoA) of words is an important psycho-linguistic property that influences various lexical tasks such as speed of reading words, naming pictures etc. In this paper, we study the effectiveness of graph theoretic and lexical features derived from dictionary networks in predicting the Age of Acquisition of English words. We show that dictionary networks contain a lot of information that can hint at the AoA of words, and the result promises that there are significant improvements over earlier approaches. We also extended the experiment to predict the Age of Acquisition of Hindi words and evaluated using words in Hindi textbooks.

1 Introduction

Age of Acquisition (AoA) (Kuperman et al., 2012) refers to the age at which a word is typically learned. For example, 'penguin' is generally learned earlier than 'albatross' and hence, 'penguin' has a lower age of acquisition value. Age of acquisition is an important feature for lexical decision tasks (Gilhooly and Logie, 1982) like speed of word reading, picture naming, word retrieval from lexical memory etc. AoA is important for two reasons. Firstly, word frequency is used as a feature alongside AoA in psycho-linguistic studies and is estimated from corpora consisting of materials meant for adult readers (Gerhand and Barry, 1999). Thus, word frequency typically underestimates the importance of words acquired earlier and thus, AoA plays an important role in complementing the information provided by word frequency. Secondly, it is believed that words acquired earlier have internal representations that can be activated faster independent of the number of times the word has been encountered. Despite the importance of the AoA parameter, the only way to estimate its value for any word is by surveying human participants. However, this process is in general tedious, requires active human participation and is thus limiting. We explore the possibility of an easier alternative. For this, we utilize the structure of word dictionary (Picard et al., 2013) where words are constructively defined.

Dictionaries provide recursive definitions and establish a dependency relation between words. To observe the inherent notion of order in which words are acquired, we consider a network constructed from a dictionary, which is a directed graph with words as vertices and edges denoting definitional participation. An edge from vertex A to vertex B implies that A is used in the definition of B in the dictionary. Thus, all predecessors of a vertex must be known for the vertex to be understood by a reader. The inherent idea is that some words (like "green") have to be learned via sensorimotor experience (Harnad, 1990), hence they are expected to be involved in loops/cycles in the network, while others can possibly be learned from constructive definitions composed of other known words which are probably simpler (Miller et al., 1990). Picard et al. (2013) extensively studied the hidden structure of the dictionary and used AoA to discriminate between different dictionary network regions successfully. This provides a strong motivation to explore the dictionary network structure for our problem - to estimate the AoA of words. In this paper, we build on this to take a step towards a faster alternative that can approximate AoA value for any word based on the dictionary network, and thus overcome the limited availability of data in addition to alleviating the need for tedious surveys.

* denotes equal contribution

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 215–223

It is important to keep in mind that the AoA is often highly subjective and can widely vary among subjects depending on their educational background, mental capabilities etc. So predicting AoA is in some way an ill-defined problem with an inherent noise in the data. Thus the primary motivation of the paper is to reveal the extent to which we can successfully predict AoA while restricting our scope to features derived from dictionary networks. It should be noted that aligning AoA may need richer cognitive and psycholinguistic features that are not contained in the dictionary network.

2 Background

2.1 Structure of Dictionary Network

Massé et al. (2008) developed a formal groundwork to determine "how word meaning is explicitly grounded in real dictionaries" and observed that meaning cannot be fetched based on dictionary definition recursively; at some point circularity of definitions must be broken by grounding the meaning of certain words. Massé et al. (2008) proposed a greedy algorithm to obtain a set of words in the dictionary network from which the rest of the words in the network can be defined without any circular dependencies. This set of words is called *grounding kernel*. The grounding kernel is estimated by repeatedly removing the nodes that do not have out-neighbors until there are no such nodes in the network. The intuition is that the nodes which do not have out-neighbors are not used for defining other words, so these words do not lead to circularity problem.

Picard et al. (2013) analyzed the full structure of the dictionary network, and they have found that the grounding kernel of dictionaries consist of a set of words, approximately 10% of the size of the entire dictionary, from which all other words can be defined. Inside the kernel, two sets of words are identified. One set is called *core* which includes the words in the strongly connected components (SCCs) that act as sources, i.e., all the words in these SCCs are defined by words in that SCC itself (in graph theoretical terms the nodes corresponding to such SCCs do not have incoming edges in its SCC-condensed graph[1]). The second set is called *satellites* which include rest of the SCCs in the grounding kernel.

[1]Each strongly connected component of a graph will be condensed to a single node in its SCC-condensed graph.

Vincent-Lamarre et al. (2016) studied the hierarchies of concepts in the concept network and proposed two types of hierarchies - kernel hierarchy and core hierarchy. These hierarchies are based on a graph theoretic property called *definitional distance*. In kernel hierarchy, the definitional distance of a word u is defined as,

1. $dist(u) = 0$, if $u \in kernel$
2. $dist(u) = 1 + \max\{dist(v) : v \in predecessors(u)\}$, otherwise

The words with definitional distance, $dist = 0$ are the words in the zeroth level of the kernel hierarchy. All words in the kernel of a dictionary will be at zeroth level. The words, which can be defined using the words in the zeroth level, are in the first level of the hierarchy and so on. For the words in the i^{th} level of the hierarchy, all the words which define these words should be at any level between 0 and $i-1$.

In core hierarchy, the definitional distance of words are computed with respect to core where core is a set of strongly connected components. The core hierarchy is a hierarchy of strongly connected components. The definitional distance of a word u from core is defined as,

1. $dist(u) = 0$, if $u \in core$
2. $dist(u) = 1 + \max\{dist(v) : \exists w \in SCC(u)$ such that $v \in predecessors(w)\}$, otherwise

where $SCC(u)$ contains all nodes in the strongly connected component that contains u.

2.2 Study of Psycho-linguistic Properties based on Dictionary Network

Vincent-Lamarre et al. (2016) studied the psycholinguistic properties such as frequency, concreteness, and age of acquisition of words with respect to the dictionary structure. This study is performed in dictionaries such as Cambridge dictionary, Longman dictionary, Merriam Webster dictionary and WordNet. Frequencies of words are computed using SUBTLEX (US) corpus; the concreteness ratings for 40,000 English word lemmas given in (Brysbaert et al., 2014) are used for concreteness; the age of acquisition ratings for 30,000 English words (Kuperman et al., 2012) are used for the age of acquisition property. They observed that the average age of acquisition of words that are part

of the core is smaller than the average age of acquisition of words that are part of satellites which is in turn smaller than the average age of acquisition of words that are not part of core and satellites. A similar trend is observed for frequency of words. For concreteness, the average concreteness value of words in satellites is observed to be smaller than the average concreteness value of words that are part of the core, which is in turn smaller than that of rest of the words. The properties such as frequency and age of acquisition follow the similar observation and it is hypothesized that words in the core are more frequent and acquired earlier compared to satellites, and words in the rest of the dictionary are less frequent and acquired later compared to satellites. This motivates us to test whether the dictionary structure will help in predicting the age of acquisition of words.

3 Dictionary Network Based Features

Let $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ be the directed graph constructed using words in the dictionary D. Each vertex $v \in \mathbb{V}$ indicates a word and each directed edge $(u, v) \in \mathbb{E}$ represents that the word u is used to define the word v in dictionary D. We propose graph-based features based on dictionary network $\mathbb{G}$ to predict the age of acquisition of each word defined in D.

We use the following basic dictionary structure-based features which were analyzed by Vincent-Lamarre et al. (2016) as having patterns related to AoA.

1. **Is core**: This is a binary value, which indicates whether the word is part of core or not. This feature is based on the work by Vincent-Lamarre et al. (2016) where they show that the words in the core are learned earlier than the satellite words, which are in turn learned earlier than the rest-of-dictionary.

2. **Is kernel**: This feature indicates whether the word is part of the kernel or not. This feature is motivated by the observation that the words in the kernel are acquired earlier than the words in the rest-of-dictionary (Vincent-Lamarre et al., 2016).

3. **Definitional distance from kernel and core**: These two features are defined by Vincent-Lamarre et al. (2016). For both definitional distances from core and kernel, Vincent-Lamarre et al. (2016) observed a linear trend between the definitional distance and the average age of acquisition of words at each definitional distance.

We propose the following features which are derived from the basic dictionary features studied by Vincent-Lamarre et al. (2016).

1. **PageRank**: The out-neighbors of a word w in $\mathbb{G}$ are the words which are defined using w. One would expect w to be acquired before words that are defined using w. Thus, *a word is acquired early if it is used to define several words that are acquired early; this leads to a circularity*. In order to resolve this circularity, we use the PageRank (Page et al., 1999) of words in the transpose graph $\mathbb{G}'$ whose vertices are same as $\mathbb{G}$, but edges are reversed to capture the importance from out-neighbors in $\mathbb{G}$.

2. **SCC PageRank**: The words within the strongly connected components (SCC) are closely associated. Because of circular dependencies between words in SCCs, it is hard to find which words are defined first. The correlation between definitional distance from core and AoA showed by (Vincent-Lamarre et al., 2016) claims that if there is an edge from the condensed node of SCC S_j to the condensed node of SCC S_i in the condensed graph, the words in S_i would be acquired earlier than words in S_j. In order to analyze the random walk interpretation of this property, we consider the importance of the SCC to which each word is associated as a feature and it is estimated as the PageRank of the corresponding condensed node in the condensed graph.

3. **Within SCC PageRank**: This feature computes the importance of a word within the SCC it belongs to as given by PageRank. We consider this feature to complement the SCC PageRank feature.

We propose the following local features to understand their trends in dictionary network for AoA prediction task.

1. **Word length**: The number of characters in the word is used as a feature. Generally, at lower ages, shorter words are learned and longer words tend to be acquired at later ages.

2. **In-degree centrality**: The in-degree centrality of a word is computed by dividing its in-degree by maximum possible in-degree. In-degree of a word w is a measure of the number of other words that are used in the definition of w. The intuition behind this is that a larger value suggests that this word may be acquired later.

3. **Out-degree centrality**: The out-degree centrality is the ratio of the out-degree of a word to the maximum possible out-degree. The out-degree of a word w is a measure of the number of words that are defined using w. The intuition is that the larger value should mean that this word may be acquired earlier.

4. **Local clustering coefficient**: The local clustering coefficient is the number of edges between immediate predecessors and successors divided by the maximum number of possible edges among them. This feature is used to study the effect of clustering tendencies on the AoA of a word.

5. **Second in-neighborhood and out-neighborhood**: The number of predecessors at distance 1 or 2 from a word in the graph is taken as the second in-neighborhood. Similarly, second out-neighborhood is the number of successors at distance 1 or 2 from a word in the graph. These features are used to study the higher level significance of in-degree centrality and out-degree centrality.

4 Experimental Evaluation

4.1 Datasets

We use the age of acquisition data from the Kuperman's (Kuperman et al., 2012) dataset and MRC psycho-linguistic dataset (Coltheart, 1981). The Kuperman dataset[2] contains AoA values for 31,124 words which are collected using Amazon Machine Turk. It contains data aggregated by asking participants to give one value in the range 1 to 25 for each word. The final AoA for a given word is then computed by taking the average of all the responses. The MRC psycho-linguistic dataset[3] contains lexical, morphological and psycho-linguistic properties of 1,50,837 words out of which the age of acquisition of 1,903 words are available. We use the age of acquisition ratings from both Kuperman and MRC datasets for evaluation.

We construct dictionary networks from Cambridge International Dictionary of English (CIDE), Longman Dictionary of Contemporary English (LDOCE), Merriam-Webster (MWC) dictionary and WordNet. We use the first definition of first sense in these dictionaries to build the corresponding dictionary graph. The stop words in the word definition are removed. The number of words in the network that is constructed from all four dictionaries are given in Table 1.

Dictionary	No of Words
CIDE	19,614
LDOCE	26,859
MWC	79,979
WordNet	76,792

Table 1: No of words in the network constructed from all four dictionaries

The graph-based features are extracted from these dictionary networks for the words that are common in all dictionaries and AoA dataset. There are 14,436 common words for Kuperman dataset, and we obtain 1,495 common words for MRC dataset.

4.2 Experiment Setup

We propose to use the dictionary network-based features along with richer cognitive and psycho-linguistic features for the prediction of age of acquisition of words. Hence, we use dictionary network-based features along with the lexical features and semantic features proposed in Paetzold and Specia (2016a) to predict the age of acquisition of words. The lexical features include

- Number of syllables
- Word's frequency in the Brown (Francis and Kucera, 1979), SUBTLEX (Brysbaert and New, 2009), SubIMDB (Paetzold and Specia, 2016b), Wikipedia, Simple Wikipedia (Kauchak, 2013) corpora
- Number of senses, synonyms, hypernyms, and hyponyms for word in WordNet

[2]Kuperman dataset is downloaded from http://crr.ugent.be/archives/806

[3]MRC psycho-linguistic dataset is downloaded from http://websites.psychology.uwa.edu.au/school/MRCDatabase/uwa_mrc.htm

- Minimum, maximum and average distance between the word's senses in WordNet and the root sense
- Number of images found for word in the Getty Image database[4]

The semantic features are the word embedded vectors (Mikolov et al., 2013) of words. The word embedded vectors capture the semantic information of words.

We train a ridge regression model to predict the AoA of words and train the model using lexical, semantic and dictionary features. We train the model using different sets of features,

1. Lexical features (Paetzold and Specia, 2016a)
2. Lexical and Semantic features (Paetzold and Specia, 2016a)
3. Lexical, Semantic and Dictionary network features based on CIDE dictionary
4. Lexical, Semantic and Dictionary network features based on LDOCE dictionary
5. Lexical, Semantic and Dictionary network features based on MWC dictionary
6. Lexical, Semantic and Dictionary network features based on WordNet

We use the lexical features and semantic features proposed in Paetzold and Specia (2016a) as a baseline to predict the age of acquisition of words. We compare the baseline model with the model that uses lexical features, semantic features, and dictionary features. We used the word embedded vectors trained using Google news dataset[5].

The model is evaluated by analyzing the Spearman's (ρ) (Spearman, 1906), Pearson's (r) (Pearson, 1920) and Kendall's tau (Sen, 1968) correlation coefficients between the actual AoA ranking and the predicted AoA ranking of words in test data. The Spearman's correlation is a measure of rank correlation and it assesses the monotonic relationships between variables. The Pearson's correlation measures the linear correlation between two variables. The Kendall's tau coefficient measures the association of variables based on pairwise ordering.

[4] http://developers.gettyimages.com/

[5] https://code.google.com/archive/p/word2vec/

4.3 Evaluation

We analyze the correlation coefficients on a 10-fold train-test splits and the average Spearman's (ρ), Pearson's (r), Kendall's tau correlations of test data are obtained. We compare the models which use dictionary network-based features with the baselines which do not use dictionary network-based features. The correlation coefficients of the predicted value with respect to the Kuperman dataset for all feature sets are listed in Table 2. All correlation coefficients are statistically significant with $p < 0.05$.

Features	Avg Spearman	Avg Pearson	Avg Kendall's tau
Lexical Features (Paetzold and Specia, 2016a)	0.4837	0.5049	0.3353
Lexical + Semantic Features (Paetzold and Specia, 2016a)	0.7793	0.7835	0.5856
Lexical + Semantic + CIDE	**0.7926***	**0.8004***	**0.5986***
Lexical + Semantic + LDOCE	0.7910*	0.7990*	0.5970*
Lexical + Semantic + MWC	0.7837*	0.7887*	0.5899*
Lexical + Semantic + WordNet	0.7878*	0.7924*	0.5941*

Table 2: Correlation Coefficients between the predicted AoA of words and the AoA based on Kuperman dataset; * indicates improvements are statistically significant with $p < 0.05$

We can observe that the model which uses dictionary network features result in better correlations compared to both baselines. The improvements in correlations are statistically significant with $p < 0.05$.

The correlation coefficients between the predicted AoA of words and the age of acquisition values in MRC psycho-linguistic dataset using all sets of features are given in Table 3. The correlation coefficients obtained when trained with all features are consistently performing better in all four dictionaries compared to baseline models.

Some words which are predicted as acquired earlier compared to its actual AoA using lexical and semantic (Paetzold and Specia, 2016a) features have their AoA predicted better when dictionary features are used. These words are observed as non-kernel words in the dictionary network. We also observed that the AoA prediction error is very less for words with a lower age of acquisition such as 'give', 'work', 'show', 'day' etc. when dictionary features are used compared to the baseline features. The inclusion of lexical and semantic features improved the prediction value for

Features	Avg Spearman	Avg Pearson	Avg Kendall's tau
Lexical Features (Paetzold and Specia, 2016a)	0.5632	0.5514	0.3993
Lexical + Semantic Features (Paetzold and Specia, 2016a)	0.7576	0.7700	0.5718
Lexical + Semantic + CIDE	0.7867*	**0.7957***	**0.6008***
Lexical + Semantic + LDOCE	**0.7880***	0.7956*	0.6005*
Lexical + Semantic + MWC	0.7609	0.7713	0.5746
Lexical + Semantic + WordNet	0.7703*	0.7820*	0.5830*

Table 3: Correlation Coefficients between the predicted AoA of words and the AoA based on MRC psycho-linguistic dataset; * indicates improvements are statistically significant with $p < 0.05$

Features	Avg Spearman	Avg Pearson	Avg Kendall's tau
Lexical Features (Paetzold and Specia, 2016a)	0.5673	0.4956	0.4004
Lexical + Semantic Features (Paetzold and Specia, 2016a)	0.7670	0.7358	0.5707
Lexical + Semantic + CIDE	0.7987*	**0.7672***	0.6026*
Lexical + Semantic + LDOCE	**0.8042***	0.7654*	**0.6068***
Lexical + Semantic + MWC	0.7741*	0.7467*	0.5783*
Lexical + Semantic + WordNet	0.7785*	0.7572*	0.5836*

Table 4: Correlation Coefficients between the predicted familiarity of words and the familiarity rating based on MRC psycho-linguistic dataset; * indicates improvements are statistically significant with $p < 0.05$

words which are i) general words like 'something', 'everyday', ii) colloquial words like 'grandma', 'mama', 'papa'. This is because of the infrequent use of such words in dictionary definitions.

We also study the prediction of other psycho-linguistic properties using dictionary network-based features. We experimented with the psycho-linguistic properties such as familiarity and imagability (Gilhooly and Logie, 1980). The familiarity of a word is the frequency with which a word is seen, heard and used. The imagability of a word is the intensity with which a word arouses images. We use the familiarity and imagability ratings presented in MRC psycho-linguistic dataset (Coltheart, 1981). This dataset contains familiarity ratings for 3,814 words and imagability ratings for 3,733 words. We trained a ridge regression model with the target value as the familiarity rating for predicting the familiarity of a word and the imagability rating is used as the target value for predicting the imagability of a word.

The same set of features are used for comparison and observed that both familiarity and imagability ratings are correlating better when dictionary network based features are used. All correlation coefficients are statistically significant with $p < 0.05$. The improvements in correlations are also statistically significant with $p < 0.05$. The dictionary features based on CIDE and LDOCE dictionaries are performing better than other two dictionaries.

Features	Avg Spearman	Avg Pearson	Avg Kendall's tau
Lexical Features (Paetzold and Specia, 2016a)	0.4734	0.4681	0.3252
Lexical +Semantic Features (Paetzold and Specia, 2016a)	0.7829	0.7708	0.5857
Lexical +Semantic + CIDE	**0.7915***	**0.7807***	**0.5939***
Lexical +Semantic + LDOCE	0.7912*	0.7797*	0.5936*
Lexical +Semantic + MWC	0.7860*	0.7757*	0.5877*
Lexical +Semantic + WordNet	0.7890*	0.7785*	0.5914*

Table 5: Correlation Coefficients between the predicted imagability of words and the imagability rating based on MRC psycho-linguistic dataset; * indicates improvements are statistically significant with $p < 0.05$

4.4 Experiment using School Textbooks

In this experiment, our task is to predict the class or grade[6] at which a word can be introduced in the school curriculum. We used the words present in Indian English textbooks from standard 1 to standard 10 which are published by National Council of Educational Research and Training (NCERT)[7]. If a word is first mentioned in i^{th} standard where $1 \leq i \leq 10$, then the target value of that word is i. In this way, we labeled the target value of words that are used in standard 1 to 10 English textbooks. We extracted all words from these textbooks and out of which we used the 5,496 words that are common in all four dictionaries.

For this task, we prefer regression over classification as the target values are ordered. We trained

[6] we use the words 'class', 'grade', or 'standard' interchangeably.

[7] http://www.ncert.nic.in/ncerts/textbook/textbook.htm

a ridge regression model to predict the target value and evaluated the model using mean squared error over 10-fold train-test splits. The model is trained using all sets of features and the results are given in Table 6. The mean squared error is highest when trained using only lexical features and it is decreased when lexical and semantic features are used. The mean squared error is even reduced when dictionary network-based features are used along with lexical and semantic features. The error is minimum when dictionary network features from LDOCE dictionary are used. The reduction in mean squared error is statistically significant with $p < 0.05$ when dictionary network-based features from CIDE, LDOCE and WordNet dictionaries are used and the deduction is statistically significant with $p < 0.01$ when MWC dictionary based features are used.

Features	Mean Squared Error
Lexical Features (Paetzold and Specia, 2016a)	0.8039
Lexical + Semantic Features (Paetzold and Specia, 2016a)	0.6621
Lexical + Semantic + CIDE	0.6326*
Lexical + Semantic + LDOCE	**0.6289***
Lexical + Semantic + MWC	0.6595+
Lexical + Semantic + WordNet	0.6553*

Table 6: Average Mean squared error when predicted the standard at which a word is first introduced to school students in NCERT English textbooks; * indicates deduction in mean squared error is statistically significant with $p < 0.05$ and + indicates deduction is statistically significant with $p < 0.01$

We also extended our experiment for Hindi words by using Hindi textbooks published by NCERT. Similar to the experiment using English Words, here the task is to predict the class or standard at which a Hindi word can be introduced in the school curriculum. We used the gloss of Hindi words in Hindi WordNet (Bhattacharyya, 2017) to construct the dictionary network using which the network based features are extracted. For lexical features, we used the frequency of words based on Hindi Corpus[8] published by Center for Indian Language Technology, IIT Bombay. The Hindi WordNet is used for other lexical features based on WordNet. We used Hindi Wikipedia[9] dump to train the embedded vectors of Hindi words. We used these features to train a ridge regression model to predict the standard at which a Hindi word can be introduced. We extracted words from NCERT Hindi textbooks from standard 1 to 10 and all features are obtained for 3,860 words. The model is evaluated using mean squared error over 10-fold train-test splits. The average mean squared error obtained when trained using i) only lexical features, ii) lexical and semantic features, iii) lexical, semantic and dictionary network features are given in Table 7. We can observe that the addition of dictionary network features improves the prediction compared to lexical and semantic features. This experiment also signifies the impact of the proposed model in predicting the age of acquisition of Hindi words for which an AoA dataset is not available.

Features	Mean Squared Error
Lexical Features (Paetzold and Specia, 2016a)	0.8708
Lexical + Semantic Features (Paetzold and Specia, 2016a)	0.8056
Lexical + Semantic + Dictionary	**0.7674***

Table 7: Average Mean squared error when predicted the standard at which a word is first introduced to school students in NCERT Hindi textbooks; * indicates deduction in mean squared error is statistically significant with $p < 0.05$

Class	Words
1	साथ (saath), काम (kaam), याद (yaad), फूल (phuul), जानवर (jaanvar)
2	आराम (aaram), दौरान (dauran), घास (ghas), असर (asar), पुलिस (pulice)
3	देखभाल (dekhbhaal), मुख (mukh), झलक (jhalak), वार (vaar)
4	दिमाग (dimaag), ट्रेन (train), जीव (jeev), उपहार (upahaar), खैर (khair)
5	पकवान (pakvaan), नौकर (naukar), बरतन (bartan), नमकीन (namkeen), मांग (maang)
6	जूट (juut), ढोलक (dholak), अधिगम (adhigam), आंगन (aangan), आश्वासन (aaswaasan)
7	गर्जन (garjan), खड्ग (khadag), जुल्म (julm), विकर्ण (vikarn), सूर्यदेव (sooryadev)
8	प्रेरक (prerak), आत्मसम्मान (aatmasamman), समाहित (samaahit), प्रतिमान (praratimaan)
9	अवमूल्यन (avamuulyan), वैष्णव (vaishnav), जागीरदार (jaageeradhaar), अध्यात्म (adhyaathm)
10	पाठ्यपुस्तक (paathyapustak), पूँजीवाद (punjeevaad), मूर्तिकार (muurtikaar), विद्याधर (vidhyaadhar)

Table 8: Examples of Hindi words predicted for Class/Standard 1 to 10

In order to qualitatively analyze the results, the predicted value of words is rounded when trained using all features. The class at which a word can be introduced is correctly predicted when the rounded predicted value is the same as the actual value assigned to the word. Some examples of Hindi words with the class/standard are correctly predicted are

[8] http://www.cfilt.iitb.ac.in/Downloads.html
[9] https://hi.wikipedia.org/

given in Table 8.

5 Discussion

Vajjala and Meurers (2014) proposed psycho-linguistic features such as the age of acquisition, familiarity, imagability and concreteness for predicting the reading level of a text and it is used for assessing the relative reading level of sentence pairs for text simplification. Our experiments suggest that the dictionary network-based features can be used as lexical features for predicting the reading level of a text.

In our experiments, the dictionary network-based features are performing better when extracted from Cambridge (CIDE) and Longman (LDOCE) dictionaries compared to Merriam Webster (MWC) and WordNet dictionaries. The motivation of proposing dictionary network-based features for predicting age of acquisition is the observation by Vincent-Lamarre et al. (2016) that the average age of acquisition of words in the core is smaller than the average age of acquisition of words in the satellites which is in turn smaller than the average age of acquisition of words in the rest of the dictionaries. In Figure 1, the average age

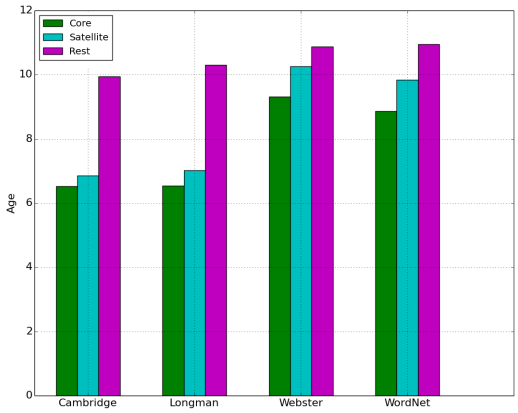


Figure 1: Average age of acquisition of words in Core, Satellites and Rest of the words in all four dictionaries

of acquisition values of words in core, satellites and the rest of the words are given for all four dictionaries. We can observe that the average AoA values are increasing from core to satellites to rest of the words in all four dictionaries. The average AoA of words in the kernel is the average of the AoA of words in core and satellites. From Figure 1, it is clear that the average AoA of kernel words will be smaller than the rest of the words. We can observe that the differences between the average AoA value of core words and the rest of the words are small in MWC and WordNet dictionaries, whereas the differences are large in CIDE and LDOCE dictionaries. This may be the reason that the CIDE and LDOCE dictionaries are performing better than MWC and WordNet dictionaries.

Concreteness (Brysbaert and New, 2009) is another psycho-linguistic property which is widely used along with age of acquisition for reading level prediction, text simplification etc. Concreteness is the extent to which the object that the word can be experienced by senses. The proposed dictionary network features are based on the dictionary structure where core words are defined before satellite words which are in turn defined before the rest of the words in the dictionary. Vincent-Lamarre et al. (2016) observed that the words present in the satellites are more concrete than the words present in the core. Since concreteness does not follow the progression from core to satellite and then to the rest of the words, which is central to our hypothesis, we did not use dictionary features to estimate concreteness.

6 Conclusion

We study the effectiveness of graph-theoretic features derived from dictionary networks in predicting the age of acquisition of words and the result shows significant improvements over earlier approaches that relate dictionary features to AoA. This work is a step towards understanding the difficult cognitive task of understanding how we acquire words. We also study the usefulness of dictionary network-based features for predicting other psycho-linguistic properties such as familiarity and imagability and the results are promising.

To the best of our knowledge, the psycho-linguistic study on Hindi has not been done before. Our experiment on Hindi words signifies the impact of the proposed model in predicting the age of acquisition of Hindi words.

References

Pushpak Bhattacharyya. 2017. IndoWordNet. In *The WordNet in Indian Languages*, pages 1–18. Springer.

Marc Brysbaert and Boris New. 2009. Moving beyond Kučera and Francis: A Critical Evaluation of Current Word Frequency Norms and the Introduction of a New and Improved Word Frequency Measure for American English. *Behavior Research Methods*, 41(4):977–990.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness Ratings for 40 Thousand Generally Known English Word Lemmas. *Behavior Research Methods*, 46(3):904–911.

Max Coltheart. 1981. The MRC Psycholinguistic Database. *The Quarterly Journal of Experimental Psychology*, 33(4):497–505.

WN Francis and H Kucera. 1979. Brown Corpus Manual. *Brown University*.

Simon Gerhand and Christopher Barry. 1999. Age of Acquisition, Word Frequency, and the Role of Phonology in the Lexical Decision Task. *Memory & Cognition*, 27(4):592–602.

Ken J Gilhooly and Robert H Logie. 1980. Age-of-acquisition, Imagery, Concreteness, Familiarity, and Ambiguity Measures for 1,944 Words. *Behavior Research Methods & Instrumentation*, 12(4):395–427.

KJ Gilhooly and RH Logie. 1982. Word Age-of-Acquisition and Lexical Decision Making. *Acta Psychologica*, 50(1):21–34.

Stevan Harnad. 1990. The Symbol Grounding Problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346.

David Kauchak. 2013. Improving Text Simplification Language Modeling using Unsimplified Text Data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1537–1546.

Victor Kuperman, Hans Stadthagen-Gonzalez, and Marc Brysbaert. 2012. Age-of-acquisition Ratings for 30,000 English Words. *Behavior Research Methods*, 44(4):978–990.

A Blondin Massé, Guillaume Chicoisne, Yassine Gargouri, Stevan Harnad, Olivier Picard, and Odile Marcotte. 2008. How is Meaning Grounded in Dictionary Definitions? In *Proceedings of the 3rd Textgraphs Workshop on Graph-Based Algorithms for Natural Language Processing*, pages 17–24. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to WordNet: An On-line Lexical Database. *International Journal of Lexicography*, 3(4):235–244.

Gustavo Paetzold and Lucia Specia. 2016a. Inferring Psycholinguistic Properties of Words. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 435–440.

Gustavo H Paetzold and Lucia Specia. 2016b. Unsupervised Lexical Simplification for Non-Native Speakers. In *AAAI*, pages 3761–3767.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford InfoLab.

Karl Pearson. 1920. Notes on the History of Correlation. *Biometrika*, 13(1):25–45.

Olivier Picard, Mélanie Lord, Alexandre Blondin-Massé, Odile Marcotte, Marcos Lopes, and Stevan Harnad. 2013. Hidden Structure and Function in the Lexicon. *arXiv preprint arXiv:1308.2428*.

Pranab Kumar Sen. 1968. Estimates of the Regression Coefficient based on Kendall's tau. *Journal of the American Statistical Association*, 63(324):1379–1389.

Charles Spearman. 1906. 'Footrule' for Measuring Correlation. *British Journal of Psychology*, 2(1):89–108.

Sowmya Vajjala and Detmar Meurers. 2014. Assessing the Relative Reading Level of Sentence Pairs for Text Simplification. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 288–297.

Philippe Vincent-Lamarre, Alexandre Blondin Massé, Marcos Lopes, Mélanie Lord, Odile Marcotte, and Stevan Harnad. 2016. The Latent Structure of Dictionaries. *Topics in Cognitive Science*, 8(3):625–659.

Author Index